

Lecture Notes in Economics and Mathematical Systems

Managing Editors: M. Beckmann and H. P. Kunzi

NCR-33-016-167

(NASA-CR-155581) LECTURE NOTES IN ECONOMICS
AND MATHEMATICAL SYSTEM. VOLUME 150:
SUPERCRITICAL WING SECTIONS 3 (New York
Univ.) 184 p HC A09/MF A01 CSCI 01A

150

N78-16018

Frances Bauer
Paul Garabedian
David Korn

Unclas
G3/02 02589

Supercritical Wing Sections III



Springer-Verlag
Berlin Heidelberg New York

Lecture Notes in Economics and Mathematical Systems

(Vol. 1–15: Lecture Notes in Operations Research and Mathematical Economics, Vol. 16–59: Lecture Notes in Operations Research and Mathematical Systems)

- Vol. 1 H. Bühlmann, H. Loeffel, E. Nievergelt, Einführung in die Theorie und Praxis der Entscheidung bei Unsicherheit 2. Auflage, IV, 125 Seiten 1969
- Vol. 2. U. N. Bhat, A Study of the Queueing Systems M/G/1 and GI/M/1 VIII, 78 pages 1968
- Vol. 3: A. Strauss, An Introduction to Optimal Control Theory. Out of print
- Vol. 4: Branch and Bound. Eine Einführung 2. geänderte Auflage Herausgegeben von F. Weinberg VII, 174 Seiten. 1973.
- Vol. 5: L. P. Hyvarinen, Information Theory for Systems Engineers VII, 205 pages-1968.
- Vol. 6: H. P. Kunzi, O. Müller, E. Nievergelt, Einführungskursus in die dynamische Programmierung IV, 103 Seiten 1968
- Vol. 7 W. Popp, Einführung in die Theorie der Lagerhaltung VI, 173 Seiten 1968.
- Vol. 8 J. Teghem, J. Loris-Teghem, J. P. Lambotte, Modèles d'Attente M/G/1 et GI/M/1 à Arrivées et Services en Groupes III, 53 pages 1969
- Vol. 9: E. Schultze, Einführung in die mathematischen Grundlagen der Informationstheorie VI, 116 Seiten 1969.
- Vol. 10. D. Hochstadter, Stochastische Lagerhaltungsmodelle VI, 269 Seiten 1969
- Vol. 11/12 Mathematical Systems Theory and Economics. Edited by H. W. Kuhn and G. P. Szegö. VIII, III, 486 pages. 1969.
- Vol. 13 Heuristische Planungsmethoden Herausgegeben von F. Weinberg und C. A. Zehnder II, 93 Seiten 1969
- Vol. 14: Computing Methods in Optimization Problems V, 191 pages 1969
- Vol. 15 Economic Models, Estimation and Risk Programming: Essays in Honor of Gerhard Tintner Edited by K. A. Fox, G. V. L. Narasimham and J. K. Sengupta VIII, 461 pages 1969.
- Vol. 16 H. P. Kunzi und W. Oettli, Nichtlineare Optimierung: Neuere Verfahren, Bibliographie. IV, 180 Seiten 1969
- Vol. 17 H. Bauer und K. Neumann, Berechnung optimaler Steuerungen, Maximumprinzip und dynamische Optimierung VIII, 189 Seiten 1969
- Vol. 18. M. Wolff, Optimale Instandhaltungspolitiken in einfachen Systemen V, 143 Seiten 1970
- Vol. 19 L. P. Hyvarinen, Mathematical Modeling for Industrial Processes VI, 122 pages 1970
- Vol. 20. G. Uebe, Optimale Fahrpläne IX, 161 Seiten 1970.
- Vol. 21: Th. M. Lieblich, Graphentheorie in Planungs- und Tourenproblemen am Beispiel des städtischen Straßendienstes IX, 118 Seiten 1970
- Vol. 22 W. Eichhorn, Theorie der homogenen Produktionsfunktion VIII, 119 Seiten 1970
- Vol. 23 A. Ghosal, Some Aspects of Queueing and Storage Systems IV, 93 pages 1970
- Vol. 24: G. Feichtinger, Lernprozesse in stochastischen Automaten V, 66 Seiten. 1970.
- Vol. 25. R. Henn und O. Opitz, Konsum- und Produktionstheorie I II, 124 Seiten 1970
- Vol. 26: D. Hochstadter und G. Uebe, Ökonometrische Methoden XII, 250 Seiten. 1970
- Vol. 27: I. H. Mufti, Computational Methods in Optimal Control Problems IV, 45 pages 1970
- Vol. 28: Theoretical Approaches to Non-Numerical Problem Solving Edited by R. B. Banerji and M. D. Mesarovic VI, 466 pages. 1970
- Vol. 29. S. E. Elmaghraby, Some Network Models in Management Science. III, 176 pages 1970.
- Vol. 30: H. Noltemeyer, Sensitivitätsanalyse bei diskreten linearen Optimierungsproblemen VI, 102 Seiten 1970.
- Vol. 31: M. Kuhlmeier, Die nichtzentrale t-Verteilung II, 106 Seiten 1970.
- Vol. 32 F. Bartholomes und G. Hotz, Homomorphismen und Reduktionen linearer Sprachen XII, 143 Seiten 1970 DM 18,-
- Vol. 33: K. Hinderer, Foundations of Non-stationary Dynamic Programming with Discrete Time Parameter. VI, 160 pages 1970
- Vol. 34 H. Störmer, Semi-Markoff-Prozesse mit endlich vielen Zuständen Theorie und Anwendungen VII, 128 Seiten 1970
- Vol. 35: F. Fersch, Markovketten VI, 168 Seiten. 1970
- Vol. 36. M. J. P. Magill, On a General Economic Theory of Motion VI, 95 pages 1970
- Vol. 37: H. Müller-Merbach, On Round-Off Errors in Linear Programming. V, 48 pages 1970.
- Vol. 38 Statistische Methoden I Herausgegeben von E. Walter. VIII, 338 Seiten 1970
- Vol. 39: Statistische Methoden II Herausgegeben von E. Walter IV, 157 Seiten 1970.
- Vol. 40. H. Drygas, The Coordinate-Free Approach to Gauss-Markov Estimation. VIII, 113 pages 1970
- Vol. 41: U. Ueimg, Zwei Lösungsmethoden für nichtkonvexe Programmierungsprobleme IV, 92 Seiten 1971
- Vol. 42 A. V. Balakrishnan, Introduction to Optimization Theory in a Hilbert Space IV, 153 pages 1971
- Vol. 43: J. A. Morales, Bayesian Full Information Structural Analysis VI, 154 pages 1971
- Vol. 44: G. Feichtinger, Stochastische Modelle demographischer Prozesse IX, 404 Seiten. 1971.
- Vol. 45 K. Wendler, Hauptaustauschschritte (Principal Pivoting) II, 64 Seiten 1971.
- Vol. 46: C. Boucher, Leçons sur la théorie des automates mathématiques. VIII, 193 pages 1971.
- Vol. 47. H. A. Nour Eldin, Optimierung linearer Regelsysteme mit quadratischer Zielfunktion VIII, 163 Seiten. 1971.
- Vol. 48 M. Constan, FORTRAN für Anfänger 2. Auflage VI, 148 Seiten 1973.
- Vol. 49: Ch. Schneeweß, Regelungstechnische stochastische Optimierungsverfahren. XI, 254 Seiten. 1971
- Vol. 50. Unternehmensforschung Heute – Übersichtsvorträge der Zürcher Tagung von SVOR und DGU, September 1970. Herausgegeben von M. Beckmann IV, 133 Seiten 1971
- Vol. 51 Digitale Simulation Herausgegeben von K. Bauknecht und W. Neß IV, 207 Seiten 1971
- Vol. 52. Invariant Imbedding Proceedings 1970. Edited by R. E. Bellman and E. D. Denman IV, 148 pages. 1971
- Vol. 53 J. Rosenmüller, Kooperative Spiele und Markte. III, 152 Seiten. 1971.
- Vol. 54 C. C. von Weizsäcker, Steady State Capital Theory. III, 102 pages 1971
- Vol. 55 P. A. V. B. Swamy, Statistical Inference in Random Coefficient Regression Models. VIII, 209 pages 1971
- Vol. 56: Mohamed A. El-Hodin, Constrained Extrema Introduction to the Differentiable Case with Economic Applications III, 130 pages 1971.
- Vol. 57. E. Freund, Zeitvariable Mehrgrößensysteme. VIII, 160 Seiten 1971.
- Vol. 58: P. B. Hagelschuer, Theorie der linearen Dekomposition VII, 191 Seiten 1971.

Lecture Notes in Economics and Mathematical Systems

Managing Editors: M. Beckmann and H. P. Künzi

150

Frances Bauer
Paul Garabedian
David Korn

Supercritical Wing Sections III



Springer-Verlag
Berlin Heidelberg New York 1977

Editorial Board

H. Albach A. V. Balakrishnan M. Beckmann (Managing Editor)
P. Dhrymes J. Green W. Hildenbrand W. Krelle
H. P. Künzi (Managing Editor) K. Ritter R. Sato H. Schelbert
P. Schönfeld

Managing Editors

Prof. Dr. M. Beckmann
Brown University
Providence, RI 02912/USA

Prof. Dr. H. P. Künzi
Universität Zürich
8090 Zürich/Schweiz

Authors

Frances Bauer
Paul Garabedian
David Korn
New York University
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, N.Y. 10012/USA

AMS Subject Classifications (1970) 76H05, 65P05, 35M05

ISBN 3-540-08533-5 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-08533-5 Springer-Verlag New York Heidelberg Berlin

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically those of translation, reprinting, re-use of illustrations, broadcasting, reproduction by photocopying machine or similar means, and storage in data banks.

Under § 54 of the German Copyright Law where copies are made for other than private use, a fee is payable to the publisher, the amount of the fee to be determined by agreement with the publisher.

© by Springer-Verlag Berlin Heidelberg 1977

Printed in Germany

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.

2142/3140-543210

PREFACE

The purpose of this book is to survey computational flow research on the design and analysis of supercritical wing sections supported by the National Aeronautics and Space Administration at the Energy Research and Development Administration Mathematics and Computing Laboratory of New York University. The work was performed under NASA Grants NGR 33-016-167 and NGR 33-016-201 and ERDA Contract EY-76-C-02-3077. Computer programs to be listed and described have applications in the study of flight of modern aircraft at high subsonic speeds. One of the codes generates cascades of shockless transonic airfoils that are expected to increase significantly the efficiency of compressors and turbines. Good simulation of physically observed flows has been achieved.

This work is a sequel to two earlier books [1,2] published by Springer-Verlag under similar titles that we shall refer to as Volumes I and II.

New York

November 1977

TABLE OF CONTENTS

I.	INTRODUCTION	1
	1. Shockless Airfoils and Supercritical Wing Sections	1
	2. Differential Equations of Gas Dynamics	2
II.	THE METHOD OF COMPLEX CHARACTERISTICS	5
	1. A New Boundary Value Problem	5
	2. Topology of the Paths of Integration	8
	3. Iterative Scheme for the Map Function	9
III.	TRANSONIC AIRFOIL DESIGN CODE	10
	1. Isolated Airfoils	10
	2. Compressor Cascades	12
	3. Turbine Cascades	13
	4. Comparison with Experiment	14
IV.	TWO-DIMENSIONAL ANALYSIS CODE	16
	1. Wave Drag	16
	2. A Fast Solver	19
	3. Remarks about Three-Dimensional Flow	24
V.	REFERENCES	26
VI.	USERS MANUAL FOR THE DESIGN CODE	29
	1. Introduction	29
	2. The Input Deck	29
	3. Closure	34
	4. Achieving a Good Design	35
	5. Boundary Layer Correction	37
	6. Error Messages	38
	7. Glossary of TAPE7 Parameters	39
	8. Glossary of Output Parameters	43
VII.	PLOTS AND TABLES OF RESULTS	45
	1. Airfoils Designed Using the New Code	45
	2. Data from Analysis and Experiment	72
VIII.	FORTTRAN LISTINGS OF THE CODES	90
	1. The New Design Code K	90
	2. Update of the Analysis Code H	167

I. INTRODUCTION

1. Shockless Airfoils and Supercritical Wing Sections

Supercritical wing technology is expected to have a significant influence on the next generation of commercial aircraft. Computational fluid dynamics has played a central role in the development of new supercritical wing sections. One of the principal tools is a fast and reliable code that simulates two-dimensional wind tunnel data for transonic flow at high Reynolds numbers (see Volume II). This is used widely by industry to assess drag creep and drag rise.

Codes for the design of shockless airfoils by the hodograph method have not been so well received because they usually require a lot of trial and error (see Volume I). However, a more advanced mathematical approach makes it possible to assign the pressure as a function of the arc length and then obtain a shockless airfoil that nearly achieves the desired distribution of pressure [11]. This tool should enable engineers to design families of transonic airfoils more easily both for airplane wings and for compressor blades in cascade.

There are plans to use the supercritical wing on commercial aircraft to economize on fuel consumption by reducing drag. Computer codes have served well in meeting the demand for new wing sections. This work is an example of the possibility of replacing routine wind tunnel tests by computational fluid dynamics.

An effective approach to the supercritical wing is through shockless airfoils. An advanced design code implementing the concept of designing a shockless airfoil so that its pressure distribution very nearly takes on prescribed data has been written recently. It has turned out to be so successful that we hope it may ultimately gain the same acceptance as the better established analysis code.

In this book we shall describe the new design code in detail and we shall give an update of the analysis code, which incorporates new

features decreasing the execution time. Fortran listings are included, together with directions for running the codes. A selection of examples and comparisons with experiment complete the work. They include shockless airfoils in cascade suggesting a new technology for turbomachinery that may contribute significantly to energy conservation.

2. Differential Equations of Gas Dynamics

The partial differential equation for the velocity potential ϕ describing isentropic flow of a compressible fluid can be derived from a variational principle asserting that the integral over the flow field of the pressure p , considered to be a function of the speed $q = |\nabla\phi|$, is stationary in its dependence on ϕ . The variational principle is useful in the formulation of finite element and finite difference methods. Here it will suffice to state the equation for ϕ in the quasilinear form

$$(c^2 - u^2)\phi_{xx} - 2uv\phi_{xy} + (c^2 - v^2)\phi_{yy} = 0$$

for plane flow, where $u = \phi_x$, $v = \phi_y$ and c is the speed of sound related to $q^2 = u^2 + v^2$ by Bernoulli's law

$$\frac{q^2}{2} + \frac{c^2}{\gamma-1} = \text{const.}$$

The normal derivative of ϕ is set equal to zero at the boundary of the flow.

For many transonic flows of practical interest, including flows around airfoils, we can assume entropy to be conserved across shock waves, so the velocity potential can be retained in the formulation of the equations of motion. Moreover, when shockless flows appear they can often be viewed as physically relevant weak solutions of the equations for which shock losses have been successfully eliminated.

We proceed to bring the equation for ϕ into a canonical form

suitable for computation. The physical coordinates x and y are connected with the velocity potential ϕ and the stream function ψ by the relation

$$x + iy = \int \frac{e^{i\theta}}{q} \left(d\phi + i \frac{d\psi}{\rho} \right) ,$$

where θ is the flow angle and ρ is the density given by the equation of state $p = \rho^\gamma$, or $c^2 = \gamma p \rho^{\gamma-1}$. We find it convenient to solve first for ϕ and ψ and obtain x and y afterwards from this integral.

The ordinary differential equation for the characteristics, or Mach lines, of the flow is known to be

$$(c^2 - u^2) dy^2 + 2uv dy dx + (c^2 - v^2) dx^2 = 0 .$$

In terms of ϕ and ψ it becomes

$$d\phi^2 + (1 - M^2) d\psi^2 / \rho^2 = 0 ,$$

where $M = q/c$ is the local Mach number. We introduce characteristic coordinates ξ and η associated with the integrals of this equation (see Volume I). In terms of ξ and η we obtain for ϕ and ψ the canonical system

$$\phi_\xi = i\sqrt{1-M^2} \psi_\xi / \rho , \quad \phi_\eta = -i\sqrt{1-M^2} \psi_\eta / \rho .$$

These partial differential equations will be integrated numerically by a finite difference scheme of the form

$$\phi_{j,k} - \tau_+ \psi_{j,k} = \phi_{j-1,k} - \tau_+ \psi_{j-1,k} ,$$

$$\phi_{j,k} - \tau_- \psi_{j,k} = \phi_{j,k-1} - \tau_- \psi_{j,k-1} .$$

Here τ_+ and τ_- stand for suitable mean values of the coefficients $\pm i\sqrt{1-M^2}/\rho$, which become known functions of ξ and η when the corresponding system of partial differential equations for the hodograph variables u and v is solved in closed form. In Volume I it has been shown how the finite difference scheme can be implemented for subsonic as well as supersonic flow through analytic extension into the complex domain.

II. THE METHOD OF COMPLEX CHARACTERISTICS

1. A New Boundary Value Problem

We have seen that physically realistic transonic flow computations can be based on partial differential equations for the velocity potential and stream function that presuppose conservation of entropy.

In terms of characteristic coordinates ξ and η we have

$$\phi_{\xi} = \tau_{+} \psi_{\xi} , \quad \phi_{\eta} = \tau_{-} \psi_{\eta} .$$

The coordinates ξ and η can be specified in terms of the speed q and the flow angle θ by the formulas

$$\log f(\xi) = \int \sqrt{1-M^2} \frac{dq}{q} - i\theta , \quad \log \overline{f(\bar{\eta})} = \int \sqrt{1-M^2} \frac{dq}{q} + i\theta ,$$

where f is any complex analytic function. Prescription of a second arbitrary function g serves to determine ϕ and ψ as solutions of the characteristic initial value problem

$$\psi(\xi, \eta_0) = g(\xi) , \quad \psi(\xi_0, \eta) = \overline{g(\bar{\eta})} ,$$

where $\xi_0 = \bar{\eta}_0$ is a fixed subsonic point in the complex plane. With these conventions it turns out that $\psi(\xi, \eta) = \overline{\psi(\bar{\xi}, \bar{\eta})}$, as can be seen from the uniqueness of the solution. Hence for subsonic flow the real hodograph plane corresponds to points in the complex domain where $\xi = \bar{\eta}$. To calculate ϕ and ψ paths of integration are laid down in the complex plane, and then a stable finite difference scheme is applied to solve the characteristic initial value problem (see Volume I).

Consider the nonlinear boundary value problem of designing an airfoil on which the speed q has been assigned as a function of the arc length s . To construct such an airfoil it is helpful to view f as a function mapping the unit circle $|\xi| < 1$ onto the region of flow. Then both $\log f$ and g have natural expansions as power series

$$\log f(\xi) = \sum a_n \xi^n, \quad g(\xi) = \sum b_n \xi^n$$

in ξ after appropriate singularities accounting for the flow at infinity have been subtracted off. The coefficients of truncations of these series can be determined by interpolating to meet boundary conditions on q and ψ at equally spaced points of the circumference $|\xi| = 1$. Such a numerical solution is easily calculated because the matrix of the system of linear equations for the coefficients is well conditioned. This analytical procedure has the advantage that its formulation can be extended to the case of transonic flow so as to yield a shockless airfoil nearly fitting the prescribed data even when an exact solution of the physical problem does not exist [11].

To calculate transonic flows by the method that has been proposed, it is necessary to circumvent the sonic locus $M^2 = 1$, which becomes a singularity of the partial differential equations for ϕ and ψ in canonical form. In the plane $\xi = \bar{\eta}$ this locus separates the region of subsonic flow from a domain where $\psi(\xi, \bar{\xi})$ is no longer real. In the latter domain it is necessary to extend in some empirical fashion the relationship between ϕ and s that is imposed by assigning q as a function of s . A formulation of the boundary conditions that applies to both the subsonic and the supersonic flow regimes is given by the formulas

$$\operatorname{Re}\{\log f(\xi)\} = \log h, \quad \operatorname{Re}\{\psi(\xi, \bar{\xi})\} + K_3 \operatorname{Im}\{\psi(\xi, \bar{\xi})\} = 0$$

on $|\xi| = 1$, where h is defined as a function of q , and therefore of $\operatorname{Re}\{\phi(\xi, \bar{\xi})\}$, by the relation

$$\log h(q) = \log h(c_*) + K_1 \left[\int_{c_*}^q |\sqrt{1-M^2}| \, dq/q \right]^{K_2},$$

and c_* is the critical speed. The real constants K_1 , K_2 and K_3 may be arbitrary for $q > c_*$, but $K_1 = K_2 = 1$ for $q < c_*$.

Empirical data on the condition number of the matrix for the

linear equations determining the power series coefficients b_n of the analytic function g indicate that the boundary value problem for ψ that has been formulated is well posed even in the transonic case [11]. In contrast with the Tricomi problem, boundary values are assigned around the whole circumference of the unit circle. The success of the procedure can be attributed to the fact that data are assigned in a suitable complex extension of the real plane. Sanz [28] has shown that the new boundary value problem for ψ is well posed in the special case of the Tricomi equation in a symmetric domain, which gives some theoretical support for the numerical method.

A sink ξ_A and a source ξ_B in the unit circle are associated with the flow at infinity for a cascade. At these points there are logarithmic singularities of the velocity potential ϕ and the stream function ψ that can be represented in the form

$$\phi = \text{Re}\{\phi_1 \log (\xi - \xi_A) + \phi_2 \log (\xi - \xi_B) + \phi_3\} ,$$

$$\psi = \text{Re}\{\psi_1 \log (\xi - \xi_A) + \psi_2 \log (\xi - \xi_B) + \psi_3\} .$$

The coefficients ϕ_j and ψ_j are regular and single-valued, and they can be calculated as solutions of characteristic initial value problems that have been described elsewhere [1,23]. They are related to the concept of the Riemann function in partial differential equations [8].

For an isolated airfoil the two points ξ_A and ξ_B coalesce and the corresponding singularity combines a pole with a logarithmic term. In all cases the singularities leave four real parameters free in the solution of the differential equations. These are to be determined through an appropriate normalization connected with the boundary conditions on q and ψ , which restrict the behavior of ϕ on the unit circle.

2. Topology of the Paths of Integration

In the design program the complex boundary value problem determining the stream function $\psi(\xi, \bar{\xi})$ in its dependence on the characteristic coordinate ξ involves a subtle choice of branch in the transonic case. The branch is specified by the paths of integration that are used in the method of complex characteristics. Over the subsonic region of flow the basic path is a radial line segment from an initial point of integration ξ_C out to the unit circle, followed by an arc of that circle. The path in the η -plane is taken to be the reflection of the path in the ξ -plane, so that real results are obtained at the diagonal points where $\eta = \bar{\xi}$.

For transonic flow the situation is more complicated because the sonic locus $M^2 = 1$, where the system of characteristic partial differential equations becomes singular, must be avoided. For supersonic paths this is achieved by using different radial stems in the ξ -plane and the η -plane, which are followed by a circular arc that is traversed in one direction on $|\xi| = 1$ while its reflection is traversed in the opposite direction on $|\eta| = 1$. For such a choice of paths $\psi(\xi, \bar{\xi})$ becomes well defined at every point on the unit circle.

The paths of integration that are used to find the real zone of supersonic flow after the boundary value problem has been solved are even more complicated. They were first introduced by Swenson [32], and one form of them is described in Volume I. For the present geometry they consist of polygonal and circular arcs connecting ξ_C to opposite ends of the arc of the sonic locus located inside the unit circle $|\xi| < 1$, followed by this arc itself, which is again traversed by ξ and $\bar{\eta}$ in opposite directions. The topology of these paths in the complex domain is such as to produce real characteristics in the physical plane corresponding to points ξ and η that vary along the sonic locus (see Figure 3, page 49).

3. Iterative Scheme for the Map Function

Consider again the question of designing an airfoil on which the speed has been given as a function of the arc length. This leads to the nonlinear boundary value problem described in Section 1 interrelating the map function f and the stream function ψ . For incompressible flow ψ can be found in closed form in the unit circle, so a linear boundary value problem for $\log f$ alone results. For compressible flow, especially in the transonic case, the question becomes more difficult and an iterative method of solution is called for.

Our procedure is first to find the map function corresponding to an incompressible flow with the prescribed pressure distribution. Given the map function, a new flow is calculated next using the method of complex characteristics to represent the regular part of the stream function ψ as a series of special solutions. When the singularities of the flow at infinity are normalized appropriately, one is led to values of ϕ or, in the transonic case, to values of $\text{Re}\{\phi(\xi, \bar{\xi})\}$, from which q can be calculated on the unit circle. This is accomplished using the known relationship between q and s , which defines q as a (multiple-valued) function of ϕ because

$$\phi(s) = \int q(s) ds .$$

The values of q lead in turn to another approximation of the map function f . We can iterate back and forth between f and ψ until a flow is determined that fits the prescribed data adequately.

The method converges extremely well for subsonic flow, presumably because the initial guess is an exact solution in the incompressible case. An exact solution cannot be expected to exist for transonic flow. However it is remarkable that a shockless flow is almost always obtained when there is one approximating the given data. Closure of the airfoil is readily attained by adjusting the pressure at the trailing edge and the relative lengths of arc over the upper and lower surfaces between the stagnation point and the trailing edge.

III. TRANSONIC AIRFOIL DESIGN CODE

1. Isolated Airfoils

Wind tunnel tests have shown that it is feasible to design high performance supercritical wing sections by solving mathematically the inverse problem of shaping an airfoil so that the transonic flow over it becomes shockless at specified conditions. The construction of shockless flows is most effectively carried out by means of the hodograph method based on complex characteristics that has been described in Chapter II. For practical applications it is essential to combine the method with a reliable turbulent boundary layer correction so as to completely suppress separation and any loss of lift associated with it.

The design method using complex characteristics is so efficient computationally that it has been tempting in the past to resolve the conflicts between various physical requirements on a desired airfoil by trial and error (see Volume I). The more systematic approach we are considering now has the advantage of eliminating the choice of many inscrutable mathematical parameters in favor of prescribing the speed q as a function of the arc length s along the profile being designed.

A problem where the specific behavior of q as a function of s becomes important is the suppression of boundary layer separation near the trailing edge of the airfoil. To treat the problem of design for transonic airfoils in a satisfactory way from the engineering point of view, it is necessary to take into account the effect of the turbulent boundary layer. A simple, direct procedure is to calculate the displacement thickness of the boundary layer from the inviscid pressure distribution by a momentum integral method. We have chosen the method of Nash and Macdonald [27]. The displacement thickness is subtracted from the airfoil coordinates, which therefore have to be

provided with a slightly open trailing edge to begin with (see Volume I).

Since it is the inviscid flow outside the boundary layer that is actually calculated, separation must be eliminated entirely in the design if there is to be no loss of lift in practice. This can be accomplished by imposing a pressure distribution near the trailing edge of the upper surface that just avoids separation according to a criterion of Stratford [31]. The essential feature of the Stratford distribution is a bound on the adverse pressure gradient (see Volume II). The boundary layer correction has been found to give satisfactory results even when its implementation only involves a primitive model of the wake in which pressure forces balance across a parallel pair of trailing streamlines. Wind tunnel test data on heavily aft loaded airfoils inspire confidence in the concept of using a Stratford pressure distribution to avoid loss of specifications in design by the hodograph method.

In the case of transonic flows calculated by the hodograph method, limiting lines may appear in the physical plane. However, it has been found that for realistic free stream Mach numbers these can be suppressed by appropriate choice of the parameters K_j that occur in our specification of the boundary conditions in Section 1 of Chapter II. Thus a tool becomes available for the construction of supercritical wing sections from their pressure distributions. Figure 2 shows an example of a shockless airfoil that was obtained this way, together with its Mach lines. Observe that the input pressure coefficient C_p differs somewhat from the values calculated as output of the flow in the supersonic zone. The data that were assigned are based on a modification of the experimental pressure distribution on Whitcomb's original supercritical wing section [33] shown in Figure 24.

The new code that has been written to implement the ideas we have described represents a major advance over what was achieved in earlier

versions (see Volumes I and II). A typical run takes about five minutes on the CDC 6600 computer. Closure is no longer a serious problem, and a valid airfoil is obtained with each run. A general principle to be observed when using shockless airfoils to design supercritical wing sections is that drag creep can be reduced by diminishing the size of the supersonic zone of flow. In practice, the best way to evaluate the performance of a new design is by means of the analysis code, which will be discussed in Chapter IV.

2. Compressor Cascades

The design code has been written to include the case of transonic airfoils in cascade. This model seems to offer considerable promise for improvement in the efficiency of certain stages of high speed compressors. However, to handle cascades of high solidity with adequate resolution it would be desirable to replace a conformal mapping onto the unit circle $|\xi| < 1$ by the mapping onto an ellipse, where the Tchebycheff polynomials become preferable to powers of ξ for expansion of the analytic functions $\log f$ and g . This is equivalent to using Laurent series expansions in a circular ring. Likewise, to achieve adequate resolution at the trailing edge in cases of heavy aft loading it would be helpful to insert a special term at the tail in the representation of the map function f .

For compressors with gap-to-chord ratio G/C not much less than unity, the present code is a very effective design tool. Since two logarithmic singularities ξ_A and ξ_B appear in the hodograph plane corresponding to the flow at infinity, more trouble is encountered with branches of logarithms along the paths of integration than is the case for isolated airfoils. However, sizeable zones of supersonic flow can be achieved near the leading edge of the cascade blades. To avoid the equivalent of drag creep, they should not be made excessively large.

If the trailing edge is handled just as in the case of isolated airfoils, no loss of specifications should be experienced in practice.. This means a Stratford pressure distribution at the rear of the upper surface is desirable. For heavy aft loading it is necessary to model the lower surface of the design calculation exactly (see Figure 8).

The concept of the supercritical compressor blade offers the possibility of reducing losses by as much as twenty percent and increasing range by a significant factor. Even for purely subsonic flow our new model of the trailing edge is significant and leads to high performance. Moreover, because realistic Reynolds numbers are 10^6 or less for compressor cascades, it is permissible to leave the trailing edge quite thick. Values of this thickness as high as two or three percent of chord are not unreasonable. Because of the lower Reynolds numbers, the transition from laminar to turbulent boundary layer plays an important role in estimation of the loss coefficient. Values a little in excess of .015 are expected.

3. Turbine Cascades

The first transonic turbine cascade with an enclosed zone of supersonic flow has been designed by McIntyre [24]. However, the present code is not altogether satisfactory for turbine cascades. Without a further version based on elliptic coordinates, gap-to-chord ratios are confined to the range $G/C > 1$. The work of Ives and Luutermoza [14] indicates that a transformation to elliptic coordinates might lower the range for G/C by a factor of two. Also, the supersonic zone cannot extend very far toward the trailing edge without violating branch cut constraints that can only be eliminated by further coding (see Section 2 of Chapter VI). For turbines this is a severe restriction because the flow accelerates as we proceed downstream. In fact, transonic turbine cascades often have exit Mach numbers above unity.

For predominantly subsonic flow our program should produce turbine blades of unusually high performance. A new feature of interest is the effect of Reynolds numbers below 10^6 , which allow for an extensive laminar boundary layer. Two examples are presented in Chapter VII.

For both compressor and turbine cascades it would be desirable to modify the design code so one could input the inlet and exit velocities to determine ξ_A , ξ_B and ξ_C automatically. The program could also be altered to adjust the input pressure distribution so that the Nash-Macdonald parameter SEP would remain constant over an arc of the airfoil near the tail (see Chapter VI). It is feasible to introduce an option to compute profiles with constant curvature over some arc near the leading edge, too. These suggestions represent a line of research that might be pursued in the future.

4. Comparison with Experiment

A program is under way to test shockless airfoils designed by the method of complex characteristics. Three-dimensional experiments have been conducted at the NASA Ames Research Center on a Boeing model of R. T. Jones' transonic transport furnished with an oblique supercritical wing we designed [2,18]. In addition to the expected improvement in performance at transonic speeds, the theoretically designed airfoil gave a seven percent increase in maximum lift-drag ratio throughout the subsonic range in a straight configuration when compared to a more conventional wing section (see Figure 30, page 87). Because transition was fixed, it is reasonable to attribute this success to our treatment of the trailing edge and the use of a Stratford pressure distribution to completely eliminate separation.

One of our compressor airfoils has been tested by the DFVLR in Germany for Harry Stephens of the Pratt and Whitney Aircraft Division of United Technologies Corporation (see Figures 8 and 31). The

transonic behavior came up to expectations. In addition, the subsonic performance was excellent over an unusually wide range. Thus the concept of supercritical airfoils in cascade offers promise of significantly increasing the efficiency of turbomachinery operating at transonic speeds.

From the experimental evidence it can now be concluded that one no longer need anticipate a loss of lift or other specifications in the design of transonic airfoils by the hodograph method, provided care is taken to eliminate separation as described in the text. Furthermore, one can suppress drag creep with the new code by spreading out the supersonic arc and working only with moderate supersonic zones safely removed from the presence of limiting lines.

IV. TWO-DIMENSIONAL ANALYSIS CODE

1. Wave Drag

Analysis of the transonic flow past an airfoil can be based on the partial differential equation

$$(c^2 - u^2)\phi_{xx} - 2uv\phi_{xy} + (c^2 - v^2)\phi_{yy} = 0$$

for the velocity potential ϕ . Weak solutions modeling shock waves are calculated by adding artificial viscosity. This can be accomplished with a full conservation form (FCF) of the equation, but a simpler version not in conservation form (NCF) is sometimes more useful [2,9,10,16,25]. To handle the boundary conditions it is convenient to map the region of flow conformally onto the interior of the unit circle and use polar coordinates r and ω there. The simplest way of introducing artificial viscosity numerically, suggested first by Murman and Cole in a fundamental paper [26], is to use finite difference approximations that are retarded in the direction of the flow. This does not perturb the Neumann boundary condition on ϕ .

The finite difference equations for transonic flow can be solved iteratively by a variety of relaxation schemes, all of which take the form of marching processes with respect to an artificial time parameter. To take viscous effects into account, a boundary layer correction is included in the computation (see Volume II). To calculate the effect of the turbulent boundary layer we actually compute the flow around a profile which results from adding the displacement thickness to the original airfoil. The displacement thickness δ and the momentum thickness θ^* are found using the method of Nash and Macdonald [27], just as in the design code. More precisely, the method is based on integration of the von Kármán momentum equation

$$\frac{d\theta^*}{ds} + (2 + H - M^2) \frac{dq}{ds} \frac{\theta^*}{q} = \frac{\tau}{\rho q^2},$$

where s , M , ρ , q and τ are arc length, local Mach number, density, speed and skin friction along the surface of the airfoil. The shape factor $H = \delta/\theta^*$ is computed by a set of semi-empirical formulas; M and q are functions of s which result from the inviscid flow computation along the profile. The ordinary differential equation for θ^* is integrated starting from fixed transition points on the upper and lower surfaces of the airfoil, where a transition Reynolds number is prescribed. Separation of the turbulent boundary layer is predicted when the Nash-Macdonald parameter

$$SEP = - \frac{\theta^*}{q} \frac{dq}{ds}$$

exceeds .004.

Detailed comparisons with experimental data show that the NCF transonic equation gives significantly better simulation of boundary layer-shock wave interaction than does the FCF equation, especially in cases with a shock at the rear of the profile where the turbulent boundary layer is relatively thick (see Section 2 of Chapter VII). It would appear that the NCF method leads to less radical gradients in the pressure behind the shock, and this is consistent with the observations [10]. The NCF and experimental speeds both tend to jump down barely below the speed of sound behind a shock. Figure 23 shows the kind of agreement between theoretical and test data that is usually seen. Wall effect is accounted for by running the computer code at the same lift coefficient C_L that occurs in the experiment.

Because of erroneous positive terms in the artificial viscosity, the shock jumps defined by the NCF method create mass instead of conserving it [10]. However, the identity

$$d(\rho q)/dq = (1 - M^2)\rho$$

shows that the amount of mass produced is only of the order of magnitude of the square of the shock strength for nearly sonic flow. The

resulting errors are therefore negligible except for their effect on the calculation of the wave drag, which has the order of magnitude of the cube of the shock strength. A correct estimate of the drag can be obtained from NCF computations by working with the path-independent momentum integral

$$D = \int [p \, dy + (\phi_x - c_*) \, d\psi] .$$

The integrand has been arranged so that across a normal shock wave parallel to the y-axis it jumps by an amount of the third order in the shock strength. Therefore integration around the shocks gives a reasonable measure of the wave drag even when mass is not conserved.

The path of integration can be deformed onto the profile to define a standard integral of the pressure there, but a correction term evaluated over a large circle should be added because of a sink at infinity accounting for the mass generated by the NCF method. Let b , ρ_∞ and q_∞ denote the chord length of the airfoil, the density at infinity and the speed at infinity, respectively. The corrected formula for the wave drag coefficient C_{DW} becomes

$$C_{DW} = \frac{2}{b\rho_\infty q_\infty^2} \int p \, dy - 2 \frac{c_* - q_\infty}{b\rho_\infty q_\infty^2} \int d\psi ,$$

where the first integral is extended over the profile and the second integral is extended over a large circle separating the profile from infinity. In Figures 26-29 a comparison is presented between experimental, corrected NCF, uncorrected NCF and FCF values of the total drag coefficient C_D for shockless airfoils tested at Reynolds number $R = 20 \times 10^6$ by Jerzy Kacprzyński at the National Aeronautical Establishment in Ottawa [19,20,21]. The corrected NCF drag formula is seen to give the most reliable assessment of the performance of the airfoils. It is incorporated in our update of the analysis code.

There are examples where the results of the NCF code agree well with experimental data right up to the onset of buffet. Shock loca-

tions are predicted with remarkable accuracy over a wide range of conditions, although some improvement would be desirable at lower Reynolds numbers where transition becomes important. Thus the analysis code has been adequately validated for simulation of experimental data in two-dimensional flow. In particular, it models the trailing edge in a satisfactory way even for heavily aft loaded airfoils. It is therefore of some interest that the code predicts no loss of lift for airfoils designed by the hodograph method when a Stratford distribution is used to eliminate separation completely over the whole profile (see Chapter III and Figure 21).

2. A Fast Solver

We have described a computer program that was developed to solve the equations of compressible flow past an airfoil in the transonic range, including a turbulent boundary layer correction (see also Volume II). The purpose of this code is to produce a reliable simulation of experimental data. A relaxation finite difference scheme is used on a grid mapped conformally onto the unit circle. The relaxation scheme has recently been modified by introducing a Poisson solver to improve the rate of convergence [17]. The Poisson solver is based on the fast Fourier transform, which is used alternately with relaxation cycles that are still needed for the stability of the scheme at supersonic points. Incorporating the Poisson solver has decreased the running time of the code on the CDC 6600 by a factor of three. This is especially important for our new drag formula (see Section 1), whose convergence rate is excessively slow using the old iterative scheme.

Since we wish to solve the transonic flow equation on a uniform grid, we use a mapping which transforms the interior of the unit circle conformally onto the exterior of the given airfoil with the origin mapped to infinity. In terms of polar coordinates, r and ω the

partial differential equation for the velocity potential ϕ becomes

$$\begin{aligned} (c^2 - \tilde{u}^2) \phi_{\omega\omega} - 2r\tilde{u}\tilde{v}\phi_{\omega r} + r^2(c^2 - \tilde{v}^2) \phi_{rr} - 2\tilde{u}\tilde{v}\phi_{\omega} \\ + r(c^2 + \tilde{u}^2 - 2\tilde{v}^2) \phi_r + r^{-1}(\tilde{u}^2 + \tilde{v}^2)(\tilde{u}\tilde{h}_{\omega} + r\tilde{v}\tilde{h}_r) = 0, \end{aligned}$$

where

$$\tilde{u} = \tilde{h}^{-1}[r\phi_{\omega} - \sin(\omega + \alpha)] , \quad \tilde{v} = \tilde{h}^{-1}[r^2\phi_r - \cos(\omega + \alpha)] .$$

Here $\tilde{h}$ is the modulus of the derivative of the mapping function multiplied by r^2 , and α is the angle of attack. The singularity of ϕ at $r = 0$ has been removed by the substitution

$$\phi = \frac{\cos(\omega + \alpha)}{r} + \Phi .$$

For a discussion of the mapping see Volume II. The logarithm of the mapping modulus $\tilde{h}$ is computed at all grid points using the fast Fourier transform to evaluate the coefficients of the power series representing it.

After the mapping is done, the velocity potential is computed at an equally spaced grid in the unit circle with coordinates r and ω , so that

$$\begin{aligned} r = j \Delta r , \quad j = 0, \dots, n; \quad \omega = \ell \Delta \omega , \quad \ell = 0, \dots, m; \\ \Delta r = \frac{1}{n} , \quad \Delta \omega = \frac{2\pi}{m} . \end{aligned}$$

In earlier work, the difference equations for Φ were solved iteratively by line relaxation. Backward differences were used in the supersonic zone and central differences at the subsonic points. Backward differencing amounts to adding a suitable artificial viscosity. At each iteration k it is convenient to solve for the change in Φ , denoted by $\delta\Phi^{(k)}$, rather than for $\Phi^{(k+1)}$ itself. This leads to an iteration of the form

$$L\delta\Phi^{(k)} = -N\Phi^{(k)} , \quad \Phi^{(k+1)} = \Phi^{(k)} + \delta\Phi^{(k)} ,$$

where N is the finite difference approximation to the differential operator on ϕ , including artificial viscosity, and L is the linear difference operator corresponding to line relaxation. Various other operators L can be introduced on the left to accelerate the method [17]. To achieve convergence, L must have an inverse and the magnitude of the largest eigenvalue of $I - L^{-1}N$ must be less than 1. This eigenvalue determines the rate of convergence. To be effective, the inversion of L must be fast and the maximum eigenvalue must be small.

In order to increase the rate of convergence of the flow calculations, a Poisson solver has been introduced to play the role of L^{-1} . For the subsonic points of the grid the terms multiplied by c^2 dominate. If we divide the differential equation for ϕ by $r^2 c^2$, these terms become

$$\Delta\phi = \phi_{rr} + \frac{1}{r} \phi_r + \frac{1}{r^2} \phi_{\omega\omega},$$

which is the Laplace operator in polar coordinates. Therefore we put

$$\Delta\delta\phi^{(k)} = R^{(k)} / r^2,$$

where $R^{(k)} = -N\phi^{(k)} / c^2$ is the residual. By substituting the values of $\phi_{lj}^{(k)}$ at the previous cycle in the difference equations, we obtain R_{lj} at each mesh point. Then we set $\phi_{lj}^{(k+1)} = \phi_{lj}^{(k)} + X_{lj}$, where X_{lj} is the correction determined by solving the discrete Poisson equation

$$A_j (X_{l,j+1} - 2X_{l,j} + X_{l,j-1}) + B_j (X_{l,j+1} - X_{l,j-1}) + (X_{l+1,j} - 2X_{l,j} + X_{l-1,j}) = R_{l,j}$$

in the unit circle, with

$$A_j = \frac{r^2 (\Delta\omega)^2}{(\Delta r)^2}, \quad B_j = \frac{r (\Delta\omega)^2}{2\Delta r}.$$

To solve for X_{lj} we introduce

$$W_m = e^{i(2\pi/m)},$$

which satisfies the orthogonality conditions

$$\sum_{k=1}^m W_m^{k\mu} W_m^{-kv} = m\delta_{\mu\nu} = \begin{cases} 0, & \mu \neq \nu, \\ m, & \mu = \nu, \end{cases}$$

for $\mu, \nu = 1, \dots, m$. Since $\delta\Phi$ is periodic in ω we write $X_{\ell j}$ as

$$X_{\ell j} = \sum_{\mu=1}^m F_{\mu j} W_m^{\ell\mu};$$

then

$$\frac{1}{m} \sum_{\ell=1}^m X_{\ell j} W_m^{-\ell\nu} = \frac{1}{m} \sum_{\mu=1}^m F_{\mu j} m \delta_{\mu\nu} = F_{\nu j}.$$

Substituting in the equation for $X_{\ell j}$, we obtain for $F_{\mu j}$ the m independent sets of tridiagonal systems

$$A_j \{F_{\mu, j+1} - 2F_{\mu, j} + F_{\mu, j-1}\} + B_j \{F_{\mu, j+1} - F_{\mu, j-1}\} - 2(1 - \cos \mu\Delta\omega)F_{\mu, j} = P_{\mu j},$$

where

$$\sum_{\mu=1}^m F_{\mu j} W_m^{\ell\mu} = R_{\ell j}.$$

From orthogonality we have

$$P_{\nu j} = \frac{1}{m} \sum_{\ell=1}^m R_{\ell j} W_m^{-\ell\nu}.$$

Thus we can solve for $P_{\nu j}$ by taking n discrete Fourier transforms of the columns $R_{\ell j}$. After obtaining the $P_{\mu j}$ we have m sets of tridiagonal systems to solve for $F_{\mu j}$. Next we take n inverse transforms of $F_{\mu j}$, which gives us the values of $X_{\ell j}$ at all grid points. The boundary conditions

$$F = 0 \text{ at } r = 0, \quad F_r = 0 \text{ at } r = 1$$

are imposed in the tridiagonal systems for $F_{\mu j}$.

In the code that implements this procedure the complex discrete Fourier transform is computed. Since all our elements are real we can obtain the transforms for two adjacent rows at a time using a method suggested by R. C. Singleton [30].

The largest eigenvalue of $I - L^{-1}N$ is bounded by M , where M is the maximum Mach number in the flow field. For purely subsonic flow the convergence rate is independent of the number of grid points. For mixed flow the method is not stable and therefore we alternate it with line relaxation steps, which are especially well suited for the supersonic zone [17].

The iterative process used in the updated analysis code can be summarized as follows:

1. The airfoil is prescribed and mapped onto the unit circle. The free stream Mach number is specified and either the coefficient of lift C_L or the angle of attack α can be prescribed.
2. The flow calculation is executed for a fixed number of cycles.
3. A new boundary layer correction is computed and added to the original airfoil to give a new profile.
4. This profile is then mapped onto the unit circle.

Steps 2 through 4 are repeated. It is reasonable to run the code for a fixed coefficient of lift because there is no other provision for wall effect.

The mapping scheme is fast and reliable, taking about two seconds of machine time on the CDC 6600. The mapping is calculated at a number of mesh points on the circle corresponding to the number of terms in the Fourier series. The mesh is chosen to give adequate resolution at the leading edge and to provide a grid size suitable for integration of the von Kármán equation for the boundary layer through a shock.

The flow calculation is done first on a crude grid in the unit circle, typically 80 intervals in ω and 15 in r . A flow cycle consists of a first pass through the points of the grid using the Poisson solver, then a fixed number NRELAX of sweeps through the grid using the relaxation scheme. We have set $NRELAX = 6$ as the default value. However this can be varied depending on the individual case to be run. Runs with small supersonic zones require fewer relaxation sweeps. After each cycle a boundary layer correction is made. We generally run 20 cycles on the crude mesh. This involves 140 passes through the finite difference grid and 19 boundary layer corrections. Before the Poisson solver was introduced we always used 800 relaxation sweeps, with a boundary layer correction after each 20 sweeps, to achieve comparable accuracy.

The iteration scheme has to be repeated on a finer grid of 160×30 mesh points. Here we generally compute 10 flow cycles, each consisting of one Poisson solver and six relaxation sweeps. The total running time for the 20 cycles on a crude grid and the 10 cycles on a fine grid, together with the necessary boundary layer corrections and mappings, is 160 seconds on the CDC 6600. This is to be compared with 520 seconds for an equivalent run without the Poisson solver. Thus the fast solver allows the convergence to proceed at a rapid rate, whereas the older relaxation scheme slowed down drastically due to a dominant eigenvalue of magnitude nearly one. The success of the fast Fourier transform depends on the fact that the number $160 = 5 \times 2^5$ of mesh points in the angle ω has many small prime factors.

3. Remarks about Three-Dimensional Flow

There is need for research on the analysis of three-dimensional transonic flow past wing-body combinations modeling an airplane. The variational principle for the velocity potential, applied in the context of the finite element method, offers the best prospect of

deriving adequate finite difference equations. Artificial viscosity must be added, and the most successful approach seems to be through an NCF formulation because of the boundary layer effect. Recently Jameson and Caughey [7] have published a swept wing code that represents substantial progress on the problem.

There is also the possibility of developing a code to design supercritical swept wings in three dimensions. For this purpose it would be desirable to combine the best features of the two-dimensional design and analysis methods we have been describing. The idea would be to assign a pressure distribution and then calculate a corresponding wing shape even in the case of transonic flow. In three dimensions the method of complex characteristics must be abandoned in favor of a scheme based on the addition of artificial viscosity in an appropriate coordinate system (see Volume II). It seems likely that through control of an artificial viscosity term smearing discontinuities adequately, flows that nearly fit a given pressure distribution and are nearly shockless might be calculated. There is little reason to suppose that designs based on such a procedure would be inferior in their overall reduction of shock losses to supercritical wings developed from perfectly shockless flow.

Mathematical tools are available to attack the kind of free boundary problem in three-dimensional space that we are concerned with here [4]. As a start in the right direction, an alternative airfoil design code should be developed that would compete with our method of complex characteristics in two dimensions (cf. [6]).

V. REFERENCES

1. Bauer, F., Garabedian, P., and Korn, D., A Theory of Supercritical Wing Sections, with Computer Programs and Examples, Lecture Notes in Economics and Mathematical Systems, Vol. 66, Springer-Verlag, New York, 1972. .
2. Bauer, F., Garabedian, P., Jameson, A., and Korn, D., Supercritical Wing Sections II, a Handbook, Lecture Notes in Economics and Mathematical Systems, Vol. 108, Springer-Verlag, New York, 1975.
3. Bauer, F., and Korn, D., "Computer simulation of transonic flow past airfoils with boundary layer correction," Proc. A.I.A.A. 2nd Computational Fluid Dynamics Conference, Hartford, Conn., June 19-20, 1975, pp. 184-204.
4. Betancourt, O., and Garabedian, P., "A method of computational magnetohydrodynamics defining stable Scyllac equilibria," Proc. National Acad. Sci. U.S.A., Vol. 74, 1977, pp. 393-397.
5. Boerstoeel, J. W., and Huizing, G. H., "Transonic shock-free aerofoil design by an analytic hodograph method," A.I.A.A. Paper No. 74-539, A.I.A.A. 7th Fluid and Plasma Dynamics Conference, Palo Alto, Calif., June 17-19, 1974.
6. Carlson, L. A., "Transonic airfoil analysis and design using Cartesian coordinates," Journal of Aircraft, Vol. 13, 1976, pp. 349-356.
7. Caughey, D., and Jameson, A., "Numerical calculation of the flow past a swept wing," ERDA Research and Development Report COO-3077-140, Courant Inst. Math. Sci., New York Univ., June 1977.
8. Garabedian, P., Partial Differential Equations, Wiley, New York, 1964.
9. Garabedian, P., "Computational transonics," Aerodynamic Analysis Requiring Advanced Computers, NASA SP-347, Part II, 1975, pp. 1269-1280.
10. Garabedian, P., "Computation of wave drag for transonic flow,"

- Journal d'Analyse Mathématique, Vol. 30, 1976, pp. 164-171.
11. Garabedian, P., and Korn, D., "A systematic method for computer design of supercritical airfoils in cascade," Comm. Pure Appl. Math., Vol. 24, 1976, pp. 369-382.
 12. Harris, A., "Two-dimensional results for airfoil M62/4," Vols. I, II, Aircraft Research Assoc. Ltd., Bedford, England, December 1974.
 13. Hicks, R. M., Vanderplaats, G. N., Murman, E. M., and King, R. R., "Airfoil section drag reduction at transonic speeds by numerical optimization," NASA TM X-73,097, February 1976.
 14. Ives, D., and Liutermoza, J., "Analysis of transonic cascade flow using conformal mapping and relaxation techniques," A.I.A.A. Jour., Vol. 15, 1977, pp. 647-652.
 15. Jameson, A., "Iterative solution of transonic flow over airfoils and wings including flows at Mach 1," Comm. Pure Appl. Math., Vol. 27, 1974, pp. 283-309.
 16. Jameson, A., "Transonic potential flow calculations using conservation form," Proc. A.I.A.A. 2nd Computational Fluid Dynamics Conference, Hartford, Conn., June 19-20, 1975, pp. 148-155.
 17. Jameson, A., "Accelerated iteration schemes for transonic flow calculations using fast Poisson solvers," ERDA Research and Development Report C00-3077-82, Courant Inst. Math. Sci., New York Univ., March 1975.
 18. Jones, R. T., "New design goals and a new shape for the SST," Astronaut. and Aeronaut., Vol. 10, 1972, pp. 66-79.
 19. Kacprzyński, J., "A second series of wind tunnel tests of the shockless lifting airfoil No. 1," Project Rep. 5X5/0062, National Research Council of Canada, Ottawa, 1972.
 20. Kacprzyński, J., "Wind tunnel test of a shockless lifting airfoil No. 2," Lab. Tech. Rep. LTR-HA-5X5/0067, National Research Council of Canada, Ottawa, 1973.
 21. Kacprzyński, J., Ohman, L., Garabedian, P., and Korn, D.,

- "Analysis of the flow past a shockless lifting airfoil in design and off-design conditions," Aeronautics Rep. LR-554, National Research Council of Canada, Ottawa, 1971.
22. Korn, D. G., "Transonic design in two dimensions," Lecture Notes in Mathematics, Vol. 430, Springer-Verlag, New York, 1975, pp. 271-288.
 23. Korn, D., "Numerical design of transonic cascades," ERDA Research and Development Report COO-3077-72, Courant Inst. Math. Sci., New York Univ., January 1975.
 24. McIntyre, E., "Design of transonic cascades by conformal transformation of the complex characteristics," ERDA Research and Development Report COO-3077-136, Courant Inst. Math. Sci., New York Univ., November 1976.
 25. Murman, E., "Analysis of imbedded shock waves calculated by relaxation methods," A.I.A.A. Jour., Vol. 12, 1974, pp. 626-633.
 26. Murman, E. M., and Cole, J. D., "Calculation of plane steady transonic flows," A.I.A.A. Jour., Vol. 9, 1971, pp. 114-121.
 27. Nash, J. F., and Macdonald, A. G. J., "The calculation of momentum thickness in a turbulent boundary layer at Mach numbers up to unity," C. P. No. 963, British A.R.C., 1967.
 28. Sanz, J., Ph.D. Thesis, New York Univ., 1978.
 29. Sells, C., "Plane subcritical flow past a lifting airfoil," Proc. Roy. Soc. London, Vol. A308, 1968, pp. 377-401.
 30. Singleton, R. C., "On computing the fast Fourier transform," Comm. A.C.M., Vol. 10, 1967, pp. 647-654.
 31. Stratford, B. S., "The prediction of separation of the turbulent boundary layer," J. Fluid Mech., Vol. 5, 1959, pp. 1-16.
 32. Swenson, E., "Geometry of the complex characteristics in transonic flow," Comm. Pure Appl. Math., Vol. 21, 1968, pp. 175-185.
 33. Whitcomb, R. T., "Review of NASA supercritical airfoils," Ninth International Congress on Aeronautical Sciences, Haifa, 1974.

VI. USERS MANUAL FOR THE DESIGN CODE

1. Introduction

This chapter is intended for users of the new design program. It is essentially independent of the material discussed earlier. After studying it readers should be able to run the program effectively.

The code, written in Fortran IV, has been run on the CDC 6600 at the Courant Institute to design isolated airfoils and compressor and turbine cascades. The isolated airfoils are easier to handle because fewer parameters are involved. A typical run at $MRP = 1$, $NFC = 32$ and $NI = 3$ (see Section 7) requires 135K octal core storage and 5 or 7 minutes CP time for an airfoil or cascade respectively.

To compute a flow the user must provide certain information, such as the free stream Mach number and flow speeds along the airfoil surface. This information is put on two data files called TAPE7 and TAPE3. These files may be either disk files or punched cards. The program computes an inviscid flow based on the input. At the discretion of the user the resulting profile can be designed to have an open tail so that a boundary layer correction can be subtracted off. The results are given as printed output and Calcomp plots.

2. The Input Deck

The input for the design program consists of two data files, TAPE7 and TAPE3. TAPE3 contains points specifying the speed q as a function of the arc length s along the airfoil measured from the lower surface trailing edge to the upper surface trailing edge. All other prescribed parameters are on TAPE7. The names and meanings of all input parameters are listed in a glossary, see Section 7.

A. TAPE7

TAPE7 uses a free form data input format, i.e. each parameter is specified by use of an 80 column punched card containing the expression

NAME=VALUE

NAME is the name of the parameter as listed in the glossary and VALUE is the numerical value assigned to it. Blanks are ignored and mode conversion (e.g. floating point to fixed point) is automatic. For a complex variable two values must be specified; they are separated by a comma. More than one parameter may be specified on a single card; then a \$ or ; must be used as a separator. The data on TAPE7 must end with a card of the form

END=

Any parameters not explicitly prescribed will take on their default values. The printed listing of TAPE7 which appears on the output is in correct format for use as input to regenerate the run.

Most of the TAPE7 parameters are easily chosen and need not be discussed. A valid flow is computed regardless of the number NI of iterations of the map function, although at least three iterations ($NI \geq 3$) are usually required to achieve an accurate fit to a given pressure distribution. Parameters such as MACH and the complex constants ξ_A , ξ_B and ξ_C require special attention.

Apart from their other functions, ξ_A and ξ_B are used to distinguish between the airfoil and cascade cases. When $\xi_A = \xi_B$, the program computes an isolated airfoil. When $\xi_A \neq \xi_B$ it computes a cascade. This is consistent with the notion that an isolated airfoil can be visualized as a cascade with an infinite gap-to-chord ratio and identical inlet and exit velocities. Throughout the manual we will discuss both situations simultaneously, allowing the context to determine which statements apply to a given case.

We now consider the hodograph plane, or complex characteristic

ξ -plane, see Figure 12, page 59. Shown on this graph are the unit circle (the subsonic portion maps onto the profile), the subsonic and supersonic integration paths, the sonic line/locus, and points of particular significance, namely the nose, the tail, the singularities ξ_A and ξ_B , the initial point for integration ξ_C , and the equally spaced mesh points along $|\xi| = 1$ used to determine the flow. The unit circle can be mapped onto itself conformally in such a way that a given pair of points, one on the boundary and one in the interior, have as their respective images a second pair of prescribed points. In practice this means that there is more than one configuration in the hodograph plane corresponding to a given flow in the physical plane. We always locate the leading edge stagnation point at $\xi = -1$, but the location of one of the singularities ξ_A and ξ_B which will yield prescribed velocities at infinity remains arbitrary. We use MACH, the Mach number corresponding to speed $q = 1$, to normalize the mapping. In general, MACH should be chosen so that the supersonic zone is of adequate size. Increasing it will raise the Mach numbers throughout the entire flow, including the free stream Mach number.

Now consider the three complex parameters ξ_A , ξ_B and ξ_C . In addition to making the distinction between isolated airfoils and cascades, ξ_A and ξ_B are important for the design of cascades because their values determine respectively the exit and inlet velocities as well as the gap-to-chord ratio. Moreover, together with ξ_C they are of critical importance for purely computational reasons, since they determine the location of two important branch cuts. The first cut, indicated in the diagram, is the line segment from ξ_A to ξ_B . The second is the ray starting at ξ_C and passing through ξ_B . The position of both cuts is of some consequence, since a supersonic integration path which inadvertently crosses one of them will cause the program to generate incorrect results and fail, terminating with a diagnostic. The isolated airfoil case is simpler because the first cut disappears

for $\xi_A = \xi_B$.

Because the program bases its choice of integration paths on the location of ξ_A , ξ_B and ξ_C , it may be helpful for the user to understand how these decisions are made. The next few paragraphs of this section give a brief explanation of that process.

Let us first consider the subsonic paths and assume ξ_A , ξ_B and ξ_C are known. To solve for the profile, the program integrates along paths which pass through ξ_A , ξ_B and ξ_C and go along the unit circle. The paths are modified to avoid crossing the branch cuts mentioned above. To achieve efficiency and accuracy, the unit circle is divided into NP arcs and an integration path is constructed for each arc.

The procedure is first to choose NP points P_i on the unit circle. Second a circle Ω of radius $|\xi_B - \xi_C|$ about ξ_B is constructed. Note that it is necessary to have ξ_C near ξ_B ; in practice we have taken $.05 \leq |\xi_B - \xi_C| \leq .15$. For each P_i on the unit circle there will be a point p_i on the circle Ω determined by the intersection of the line from P_i to ξ_B . An arc opposite ξ_C on Ω is omitted to avoid the branch cut from ξ_C through ξ_B . Thus each subsonic path consists of (1) the polygonal line from ξ_A to ξ_B to ξ_C , (2) an arc along Ω from ξ_C to p_i chosen to avoid the branch cut, (3) the line segment from p_i to P_i and (4) an arc on the unit circle from P_i to P_{i+1} .

A few comments are now in order. The choice of paths is arranged to insure that, without placing unnecessary restrictions on the choice of ξ_A and ξ_B , we either avoid crossing the branch cuts or, being aware of their position, correct the computation when we do so. The necessary corrections are made to the computation for subsonic paths. Furthermore, any of the line segments between p_i and P_i will be deformed if they come too close to ξ_A .

Arcs of the unit circle outside the subsonic domain are still used in the computation, but they are distinguished by two asymmetric

polygonal lines connecting them to the circle Ω . These supersonic paths are more troublesome because the code has not been designed to take the branch cuts into consideration. The computation terminates with a diagnostic if such a path crosses a cut.

In choosing the paths the points at which the sonic line intersects the unit circle are determined. When there are no such intersections, i.e. when MACH has a low value, the flow is entirely subsonic and supersonic paths are not needed. If MACH becomes higher, one or more supersonic zones may appear. Good resolution is only obtained when these zones are large enough to be well-defined numerically, so results are poor for flow that is just barely supersonic. Calculation of the supersonic portion of the flow is based on a pair of asymmetric supersonic paths pictured in the diagram that contain arcs of the circle $|\xi| = 1.04$.

It is up to the user to choose ξ_A , ξ_B , ξ_C and MACH so that the supersonic paths do not cross any cuts, although the program may itself deform the paths to avoid singular points at which $M^2 = 1$. It is often helpful to start with a low value of MACH, thus eliminating the supersonic paths altogether until ξ_A , ξ_B and ξ_C are properly positioned. Perhaps the best way to avoid crossing cuts unnecessarily is to allow ξ_C to assume its default value. If this does not suffice, another possibility is to input a pure imaginary value of ξ_C , which instructs the code to place ξ_C at a point on the circle of radius $|\text{Im}\{\xi_C - \xi_B\}|$ around ξ_B such that ξ_A , ξ_B and ξ_C become colinear and all cuts coincide.

We observe that in the design code the supersonic paths can be altered so as to extend the range of the method by changing the calls to the subroutine PATH from the subroutine GTPATH.

B. TAPE3

The data on TAPE3 appears in two formats. The first card has the value of NIN, the number of points defining the input speed distribution or, more simply, the number of cards to follow on this data file. NIN should be a right justified integer constant appearing in the first five columns of the card (Fortran format (F5.0)).

The remaining NIN cards contain the arc length s in columns 1-20 and the corresponding flow speed q , or its negative, in columns 21-40 (Fortran format (2E20.8)). Note that q is input negative for points of the lower surface and positive for upper surface points. Also, it is assumed that s runs along the airfoil monotonically increasing from the lower surface tail to the upper surface tail. However, the user may choose any values of s he wishes for the beginning and end points. The program will rescale the coordinates so that s varies from -1 to +1.

In general the input data for TAPE3 will depend on the particular airfoil the user intends to design. Sometimes a minor difficulty arises when spline interpolation results in a bad fit to the data. This can be overcome by inserting more data points in the region of difficulty.

Modifications of the input data on TAPE3 can also be made by means of bump functions described in the glossary. They facilitate changing the pressure distribution at a few points without introducing undesirable oscillations that often occur in the splines defined by the code.

3. Closure

As we mentioned earlier, the user may choose to design an airfoil with an open trailing edge, as in Figure 13, page 60. In that case we measure the amount of separation of the trailing edge by the quantities DX and DY , the respective differences between the x and y coordinates of the upper and lower surface end points. In general it

is desirable that after a boundary layer correction has been made the separation be about .7% of chord for an isolated airfoil and 2% of chord for a cascade. In any case, DX and DY should be chosen so that the line segment connecting the two end points at the trailing edge is perpendicular to both surfaces there. We note that, as might be expected, the opening of the trailing edge is coupled to the thickness-chord ratio.

Adjusting either of the parameters DX and DY is a rather straightforward process. In fact, we can raise or lower DX by simply rescaling s on the upper surface or the lower surface separately. In other words, too large (or too small) a DX indicates that there is too much (or too little) arc length on the upper surface as opposed to the lower surface. For small discrepancies, we can remedy the situation by decreasing (or increasing) the s coordinate at the last few points of the upper surface input speed distribution (TAPE3) or by doing the opposite to the first few points from the trailing edge on the lower surface.

DY, on the other hand, may be raised or lowered by lowering or raising the pressure coefficient C_p at the trailing edge, which in practice means raising or lowering q at the first and last points of the input speed distribution (TAPE3). It may become necessary, however, to move several values on the upper or lower surface as a unit in order to control boundary layer separation.

4. Achieving a Good Design

We shall describe a collection of techniques found useful in designing airfoils in the past. Several parameters not related to the boundary layer correction will be discussed here, although we have made no attempt to be exhaustive. It is assumed that the user already has some knowledge of the relationship between specific design constraints and the speed function along the airfoil surface. We observe

that such relationships are similar to those for purely subsonic flow. It is important to realize that many quantities are coupled, so that correcting one situation may cause difficulties with another. The user will have to learn which parameters to adjust first and at which stages in the design process to accept crude approximations to the ultimate goal.

Suppose we are given the inlet and exit Mach numbers M_1 and M_2 and the inlet and exit flow angles θ_1 and θ_2 . These parameters are determined, for a fixed value of MACH, by the positions of ξ_B and ξ_A , which may be set in the following way. First the user should locate ξ_A so that it yields roughly the desired M_2 and θ_2 but is not too close to the unit circle. An important tool in this process is the plot of the curves of constant Mach number (see IPLT in Section 7). Next, for fixed ξ_A search for the position of ξ_B which corresponds to M_1 and θ_1 . Note that it is sometimes easier to start with a relatively low value of MACH so as to eliminate the supersonic paths. Then, after placing ξ_B so that $\theta_2 - \theta_1$ and $M_2 - M_1$ are correct, we may bring M_1 and θ_1 near their desired values by raising MACH to a higher value. The gap-to-chord ratio G/C is also involved in these considerations, since it depends on the magnitude of $|\xi_B - \xi_A|$. Thus we can raise G/C by bringing ξ_A and ξ_B closer together, which has an obvious effect on the velocities at infinity.

The coefficient of lift C_L is directly related to the area enclosed by the pressure distribution curves. We can raise or lower the lift by moving the input speed distribution curves of the upper and lower surfaces farther apart or closer together. As we have observed before, the thickness-chord ratio T/C is coupled to the tail separation. However, it has a much stronger relationship to the radius of curvature of the leading edge, which itself depends on the derivative $q'(s)$ of the input speed distribution at $q = 0$. Thus we can lower T/C by increasing the slope of the input curve $q(s)$ near the

leading edge stagnation point.

5. The Boundary Layer Correction

Since we have based the design program on inviscid theory, it is necessary to make a boundary layer correction to allow for viscous effects. This is done by assuming that the computed streamlines delineate, not the airfoil itself, but the flow outside the boundary layer. We then use the von Kármán momentum equation to compute the displacement thickness of the boundary layer, which we subtract off from the previously computed inviscid coordinates of the airfoil. No laminar boundary layer correction is made; instead, points of transition are assigned and the turbulent boundary layer method of Nash and Macdonald is followed. This correction is suppressed in the code by setting $RN = 0$.

We emphasize how important it is in practice that the boundary layer not separate. To avoid separation we require that the Nash-Macdonald parameter SEP stay below the bound

$$SEP \leq .004 .$$

In practice we have found it desirable to have SEP remain approximately constant and near .003 along the upper surface close to the trailing edge. This can be achieved by adjusting the slope of the speed there. Airfoils satisfying this requirement generally meet their design specifications in wind tunnel testing and provide excellent performance at off-design conditions as well.

Implementing the boundary layer correction is quite simple. Aside from the Reynolds number RN, the user must specify TRANU and TRANL, the x coordinates along the upper and lower surfaces at which transition occurs, i.e. where the correction is to begin. The values of TRANU and of TRANL are chosen empirically. After the calculation is completed, both inviscid and corrected x, y coordinates of the airfoil are plotted and printed together with SEP and the momentum thickness

THETA.

6. Error Messages

Below we give a complete list of error messages contained in the code and, where feasible, suggest procedures to use when they occur. Since most of the trouble arises in supersonic cases, a lower choice of MACH is often helpful when a run has terminated with a diagnostic.

1. ATTEMPT TO CONTINUE RUN _____ WITH WRONG RUN NUMBER
2. AUTOMATION PATH IS TOO LONG
Remedy: Lower MRP or increase GRID.
3. ERROR IN CARD _____
Remedy: Self-explanatory.
4. MORE THAN _____ INPUT CARDS NOT PERMITTED
PROGRAM STOPPED IN READQS
Remedy: User is limited to 300 data points, but is NIN too small?
5. NEWTON ITERATION DID NOT CONVERGE AT ST= _____
6. NK = _____ AND NT = _____ ARE INCOMPATIBLE
PROGRAM STOPPED IN CYCLE
7. NO CONVERGENCE AT S= _____ XI= _____ S(XI) -S= _____
TROUBLE CALCULATING SONIC LINE, CHANGE MACH
8. NON-ALPHABETIC CHARACTERS NOT ALLOWED IN VARIABLE NAMES
9. NON-MONOTONIC ABSCISSA FOR SPLINE FIT
PROGRAM STOPPED IN SUBROUTINE PSPLIF
10. NON-MONOTONIC ABSCISSA FOR SPLINE FIT
PROGRAM STOPPED IN SUBROUTINE SPLIF
11. AUTOMATED SUPERSONIC PATH CROSSES CUT FROM XIA TO XIB
Remedy: The sector of automation paths enclosing ξ_A must not intersect the sonic line.
12. AUTOMATED SUPERSONIC PATH CROSSES CUT FROM XIC THROUGH XIB
Remedy: Try default value of XIC or pure imaginary option.

13. SUPERSONIC PATH IS TOO LONG

Remedy: Lower MRP or increase GRID.

14. TOO MANY POINTS DEFINING THE AIRFOIL

PROGRAM STOPPED IN BODYPT

15. TOO MANY SUPERSONIC POINTS

16. TRANSITION NOT FOUND -- BOUNDARY LAYER SKIPPED

17. VARIABLE NAMED WAS NOT FOUND

18. WT DID NOT CONVERGE, WT = ____

19. NEWTON ITERATION FOR SONIC POINT DID NOT CONVERGE WELL

7. Glossary of TAPE7 Parameters

All TAPE7 input parameters are listed below with the proper name, default value, definition, and usage. The order corresponds to the frequency of use in the examples we ran. In certain cases we have written DEFAULT VALUE RECOMMENDED. The user is strongly advised to use the default values for those parameters, especially if there is any difficulty.

Name	Default	Definition	Usage
RUN	1	Run number	Abs(RUN) is the run number, an identification number for plotted and printed output. A negative value of RUN will produce a white paper plot on the CDC 6600 at the ERDA Mathematics and Computing Laboratory at New York University.
NI	1	Number of iterations of map function	Number of iteration cycles. If $NI \leq 0$, no flow cycles will be computed, but the s and q input will be plotted.
MRP	-1	Mesh refinement parameter	Controls the number of points on the grid, e.g. doubling it will cut the mesh spacing in half. If $NI > 1$, an increase in MRP will only be done on the last iteration. MRP negative gives third order accuracy by use of a Richardson extrapolation. Thus, for example, $MRP = -2$ will require as much core storage as $MRP = 4$.

Name	Default	Definition	Usage
MACH	.75	Mach number at speed $q = 1$	Determines the critical speed. MACH can be used to increase or decrease the size of the supersonic zone. It is related, but not equal, to the free stream Mach number.
XIA	0.08, -.12	ξ_A , location of logarithmic singularity determining exit Mach number and angle of flow	The point in the ξ -plane corresponding to the exit velocity (i.e. the hodograph image of $x = +\infty$). To get an isolated airfoil, set $XIA=XIB$. For cascades XIA should lie inside a sector of automation paths which does not meet the sonic line. DEFAULT VALUE RECOMMENDED FOR ISOLATED AIRFOILS.
XIB	0.08, -.12	ξ_B , location of logarithmic singularity determining inlet Mach number and angle of flow	The point in the ξ -plane corresponding to the inlet velocity (i.e. the hodograph image of $x = -\infty$). To get an isolated airfoil, set $XIB=XIA$. DEFAULT VALUE RECOMMENDED FOR ISOLATED AIRFOILS.
XIC	If $XIA=XIB$, $XIC=XIA$ -.1-.1i; if $XIA \neq XIB$, $XIC=XIB-.07$ $-(XIA-XIB)$ $/ XIA-XIB $	ξ_C , point determining characteristic initial plane	Point of initiation of paths of integration. If XIC is pure imaginary then for $XIA=XIB$, $XIC = XIA - Im\{XIC-XIA\} e^{i\pi/4}$; and for $XIA \neq XIB$, $XIC = XIB - Im\{XIC-XIB\} (XIA-XIB)/ XIA-XIB $. DEFAULT VALUE RECOMMENDED.
RN	0.	Reynolds number of the flow	Set $RN = 0$ to suppress the boundary layer correction.
TRANU	.05	Abscissa for transition on upper surface	The value of x (before rotation) at which the turbulent boundary layer integration on the upper surface starts.
TRANL	.10	Abscissa for transition on lower surface	The value of x (before rotation) at which the turbulent boundary layer integration on the lower surface starts.

Name	Default	Definition	Usage
IPLT	37	Index for plot control	A two digit number which controls the plotting. The first digit controls the plot of the physical plane: 0 no plot generated 1 airfoil and Mach distribution 2 same as 1 with airfoil higher 3-4 same as 1-2 with characteristics 5-9 same as 0-4 but with pressure plot instead of Mach plot. The last digit controls the plot of the ξ-plane: 0 no plot generated 1-4 sonic locus, nodes on circle, points ξ_A, ξ_B, ξ_C, nose, tail and integration paths are plotted 5-9 same as 0-4 with contour curves $q = \text{constant}$.
PLTSZ	20.	Plot size	Size in inches of the large airfoil plot. If PLTSZ=0, no plot will be generated. If PLTSZ<0, no symbols will be plotted at calculated points.
NFC	64	Number of Fourier coefficients	The degree of the polynomial part of the mapping function. It should not exceed NF/2. DEFAULT VALUE RECOMMENDED.
NF	0	Number of functions	The number of functions in the series representing ϕ and ψ. Equal to the number of nodes on the circle $\xi =1$. If NF<0, NF will be set to 2NFC. DEFAULT VALUE RECOMMENDED.
NP	8	Number of automation paths	The unit circle is divided into NP arcs for the integration. NP must be an even integer that divides NFC and does not exceed 16. DEFAULT VALUE RECOMMENDED.
GRID	.08	Grid spacing parameter	GRID divided by MRP is the maximum mesh size. DEFAULT VALUE RECOMMENDED.
BUMP	None	Bump vector with five components separated by commas	The vector BUMP=FMAX,XL,XR,RAT,ALP is used to modify the input speed $q(s)$ in the interval $XL < s < XR$ by the factor $1 + FMAX[1 - T^2]^{ALP}$, where $T = \frac{s - XL - RAT(XR - XL)}{(1 - 2RAT)(s - XL) + RAT(XR - XL)}$ Ten such bumps are allowed.

Name	Default	Definition	Usage
GAMMA	1.4	Gas constant	GAMMA is 1.4 for air, and 1.07 for uranium hexafluoride.
NPTS	201	Number of points	Number of nodes in the spline defining the input distribution. DEFAULT VALUE RECOMMENDED.
KONE	.5	K_1	Used along with K_2 to extend $h(q)$. DEFAULT VALUE RECOMMENDED.
KTWO	1.0	K_2	Used along with K_1 to extend $h(q)$. DEFAULT VALUE RECOMMENDED.
KTHR	1.0	K_3	Used for boundary condition along the transonic arc of the circle. DEFAULT VALUE RECOMMENDED.

8. Glossary of Output Parameters

All output parameters are listed below by symbol and name and, where different from common practice, the particular way in which they have been defined in the program is specified.

Symbol	Name	Meaning
ANG	Flow angle	Angle of tangent along airfoil.
C_L	Coefficient of lift	Standard lift coefficient.
DEL TH	Turning angle	The change in flow angle from $x = -\infty$ to $x = +\infty$.
Diffusion factor	Diffusion factor	Standard measure of performance of compressor blades.
DX	DX	The change in x coordinate (before boundary layer correction) from the lower surface to the upper surface at the trailing edge.
DY	DY	The change in y coordinate (before boundary layer correction) from the lower surface to the upper surface at the trailing edge.
Exit flow angle	Exit flow angle	The clockwise angle between the vertical and the flow at $x = +\infty$.
G/C	Gap-to-chord ratio	Vertical distance between blades divided by chord length.
Inlet flow angle	Inlet flow angle	The clockwise angle between the vertical and the flow at $x = -\infty$.
K	Curvature	Curvature of the airfoil at a given point (before the boundary layer correction).
Loss coefficient	Loss coefficient	Loss coefficient computed from boundary layer correction.
M	Free stream Mach number	Mach number at infinity.
M_1	Inlet Mach number	Mach number at $x = -\infty$.
M_2	Exit Mach number	Mach number at $x = +\infty$.
Profile drag coefficient	Profile drag coefficient	Drag coefficient computed from boundary layer correction.

Symbol	Name	Meaning
SEP	Nash-Macdonald separation parameter	$SEP = - (\theta^* dq) / (q ds)$. The boundary layer is predicated to separate for $SEP \geq .004$.
T/C	Thickness-to-chord ratio	Difference of ordinates divided by difference of abscissas.
THETA	Momentum thickness θ^*	Momentum thickness of the boundary layer.
TRANSITION	Transition point	Point on the upper (or lower) surface at which the turbulent boundary layer correction begins.
X	Horizontal airfoil coordinate	x coordinate of the airfoil before the boundary layer correction.
XS	Corrected horizontal airfoil coordinate	x coordinate of the airfoil after the boundary layer correction.
Y	Vertical airfoil coordinate	y coordinate of the airfoil before the boundary layer correction.
YS	Corrected vertical airfoil coordinate	y coordinate of the airfoil after the boundary layer correction.

VII. PLOTS AND TABLES OF RESULTS

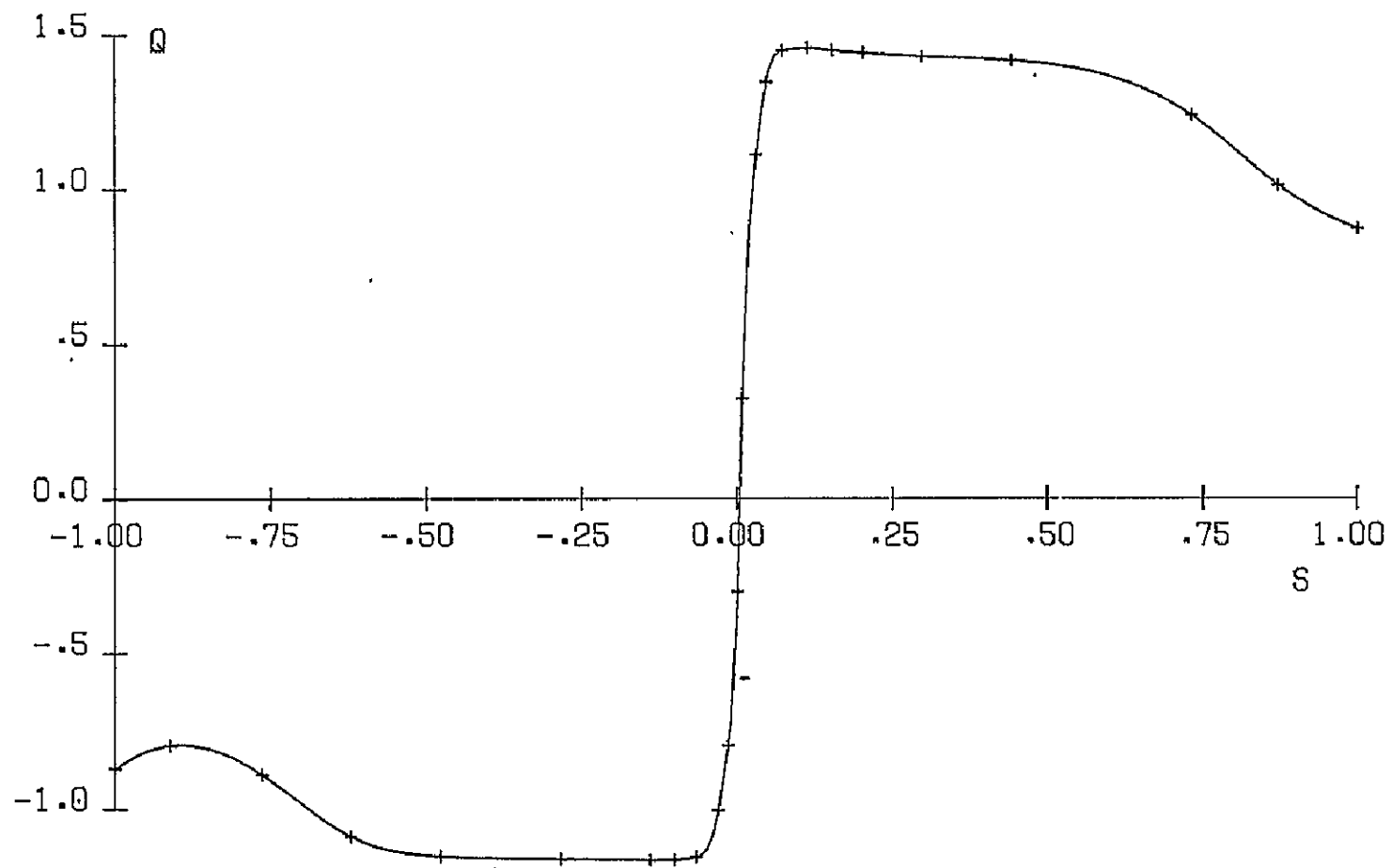
1. Airfoils Designed Using the New Code

Below we present several airfoils that were designed by use of the new code K.

A. Figures 1-7

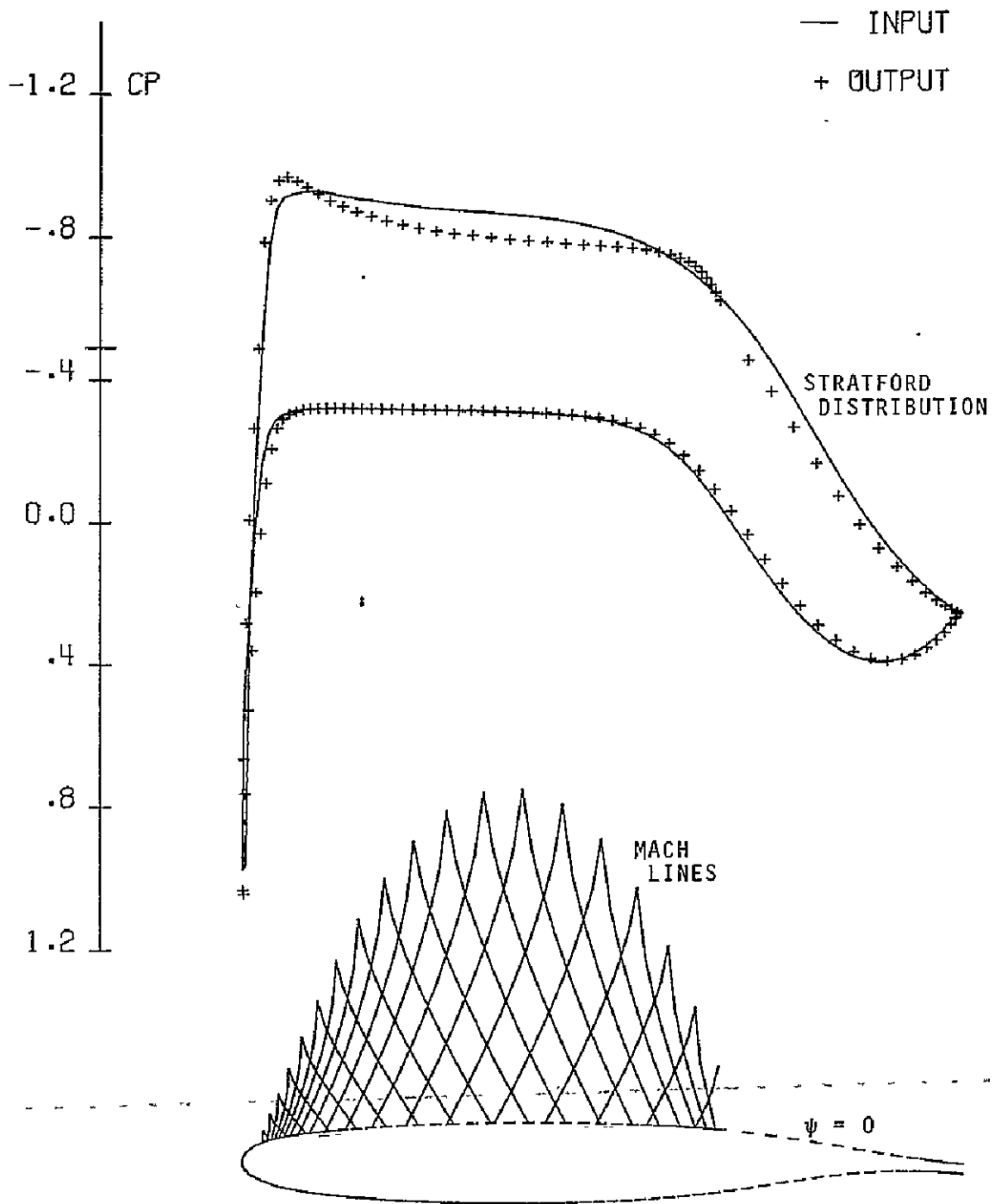
Figure 2 represents a modification of the original supercritical wing section designed empirically by Whitcomb [33]. The input for $q(s)$ shown in Figure 1 was developed from the experimental pressure distribution presented in Figure 24. We have eliminated much of the aft loading by shifting the lower surface pressure curve in the direction of higher speeds, thus decreasing the area enclosed. Also, the upper surface pressure distribution has been altered to meet the Stratford criterion. The shock that appeared in the experiment was eliminated simply by omitting several subsonic data points. The complex characteristic hodograph plane for the shockless design calculation is presented in Figure 3.

Figures 4 and 5 show a case designed for a rather low lift coefficient. Figures 6 and 7 present the case of a perfectly symmetric airfoil with two supersonic zones and zero lift. In these illustrative examples it is important to look at the geometry of the Mach lines in the physical plane and the integration paths in the hodograph plane. We note that if the parameter MACH is gradually increased while the rest of the input is held fixed, limiting lines begin to appear in the flow and then later the supersonic paths of integration cross branch cuts causing the code to fail (see Section 2 of Chapter VI).



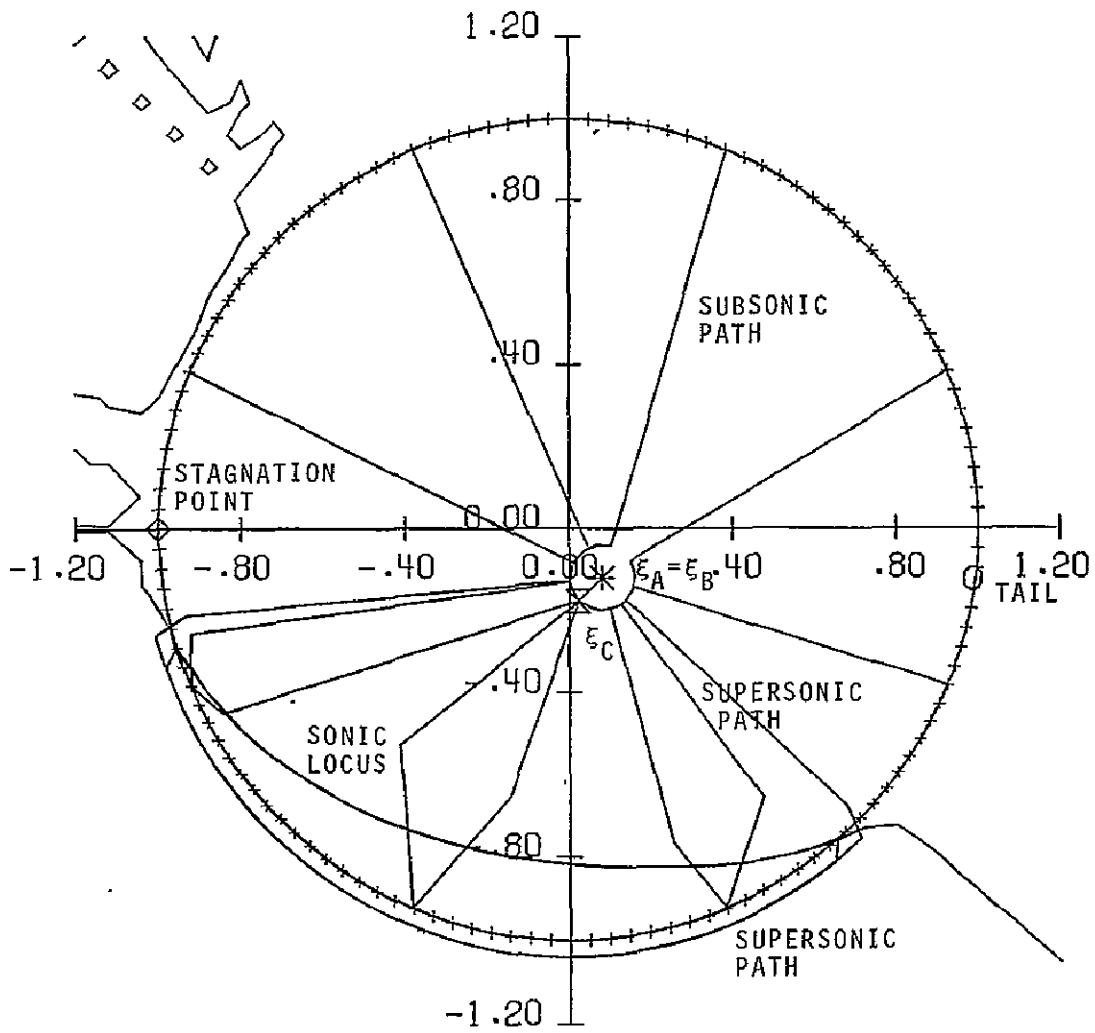
INPUT SPEED DISTRIBUTION

FIGURE 1. INPUT $q(s)$ FOR MODIFIED WHITCOMB WING



$M = .781$ $CL = .479$ $DX = .001$ $DY = .014$ $T/C = .113$

FIGURE 2. MODIFIED WHITCOMB WING SECTION



$M = .781$ $CL = .479$ $DX = .001$ $DY = .014$ $T/C = .113$

FIGURE 3. COMPLEX CHARACTERISTIC HODOGRAPH PLANE

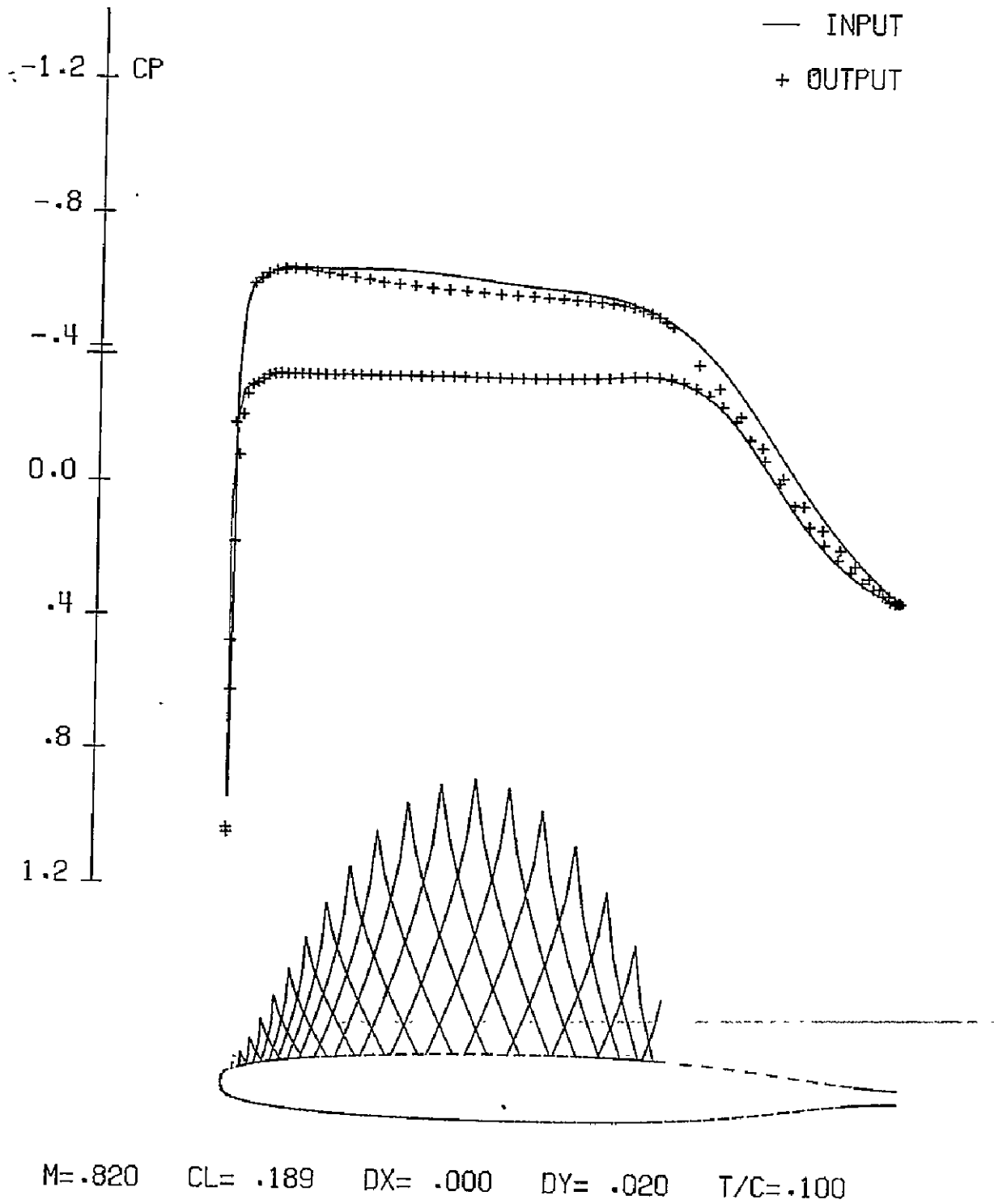
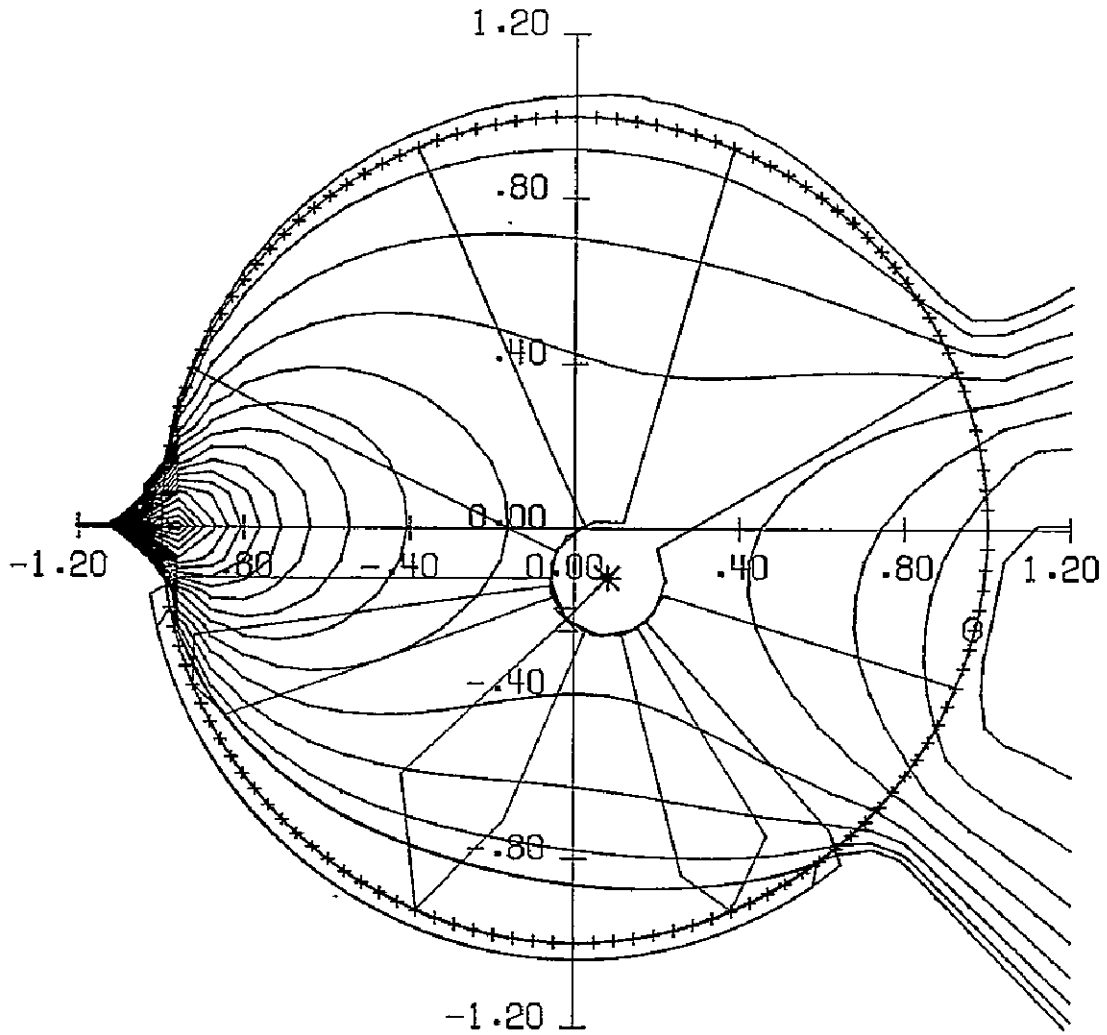
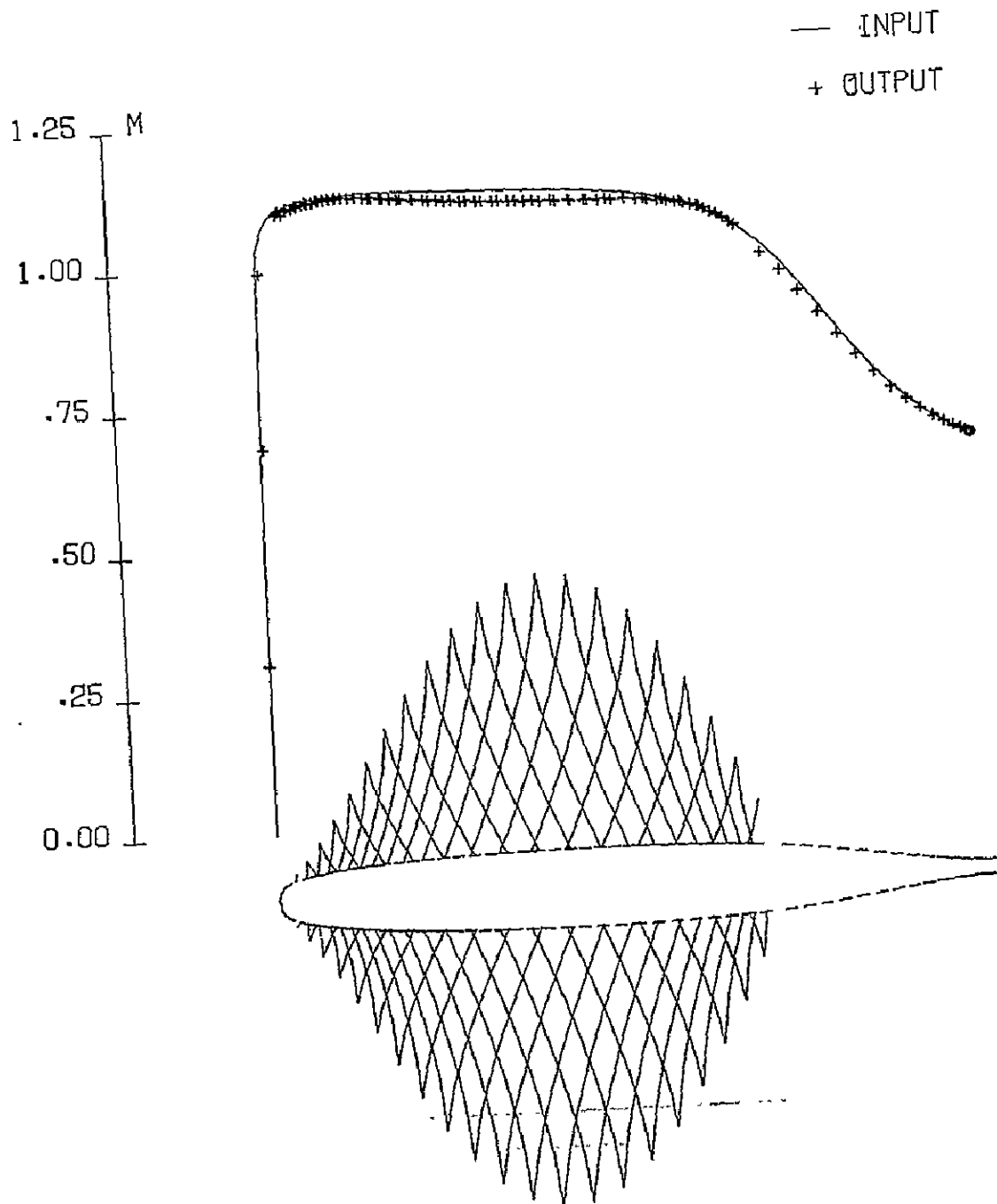


FIGURE 4. LOW LIFT SHOCKLESS AIRFOIL



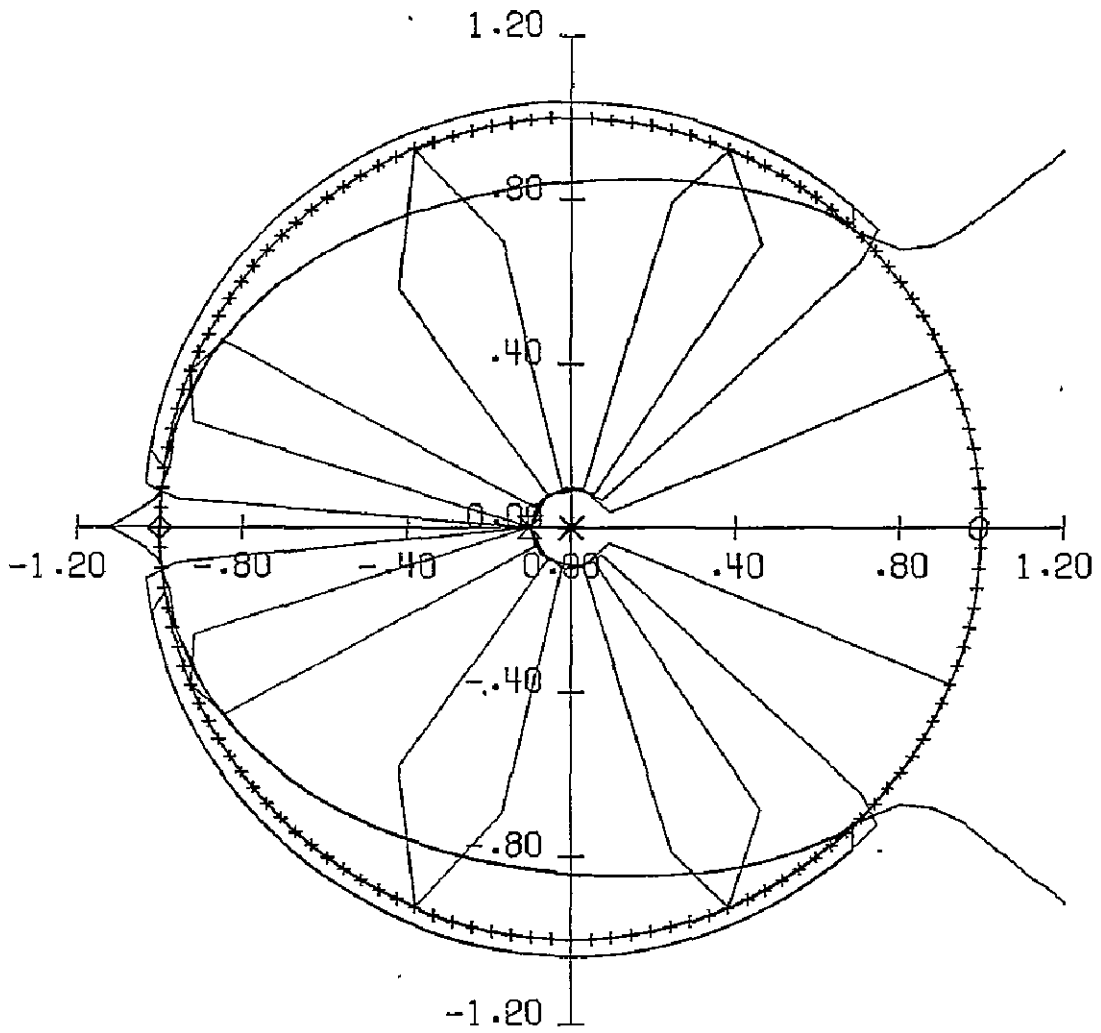
$M=.820$ $CL=.189$ $DX=.000$ $DY=.020$ $T/C=.100$

FIGURE 5. HODOGRAPH PLANE WITH LEVEL CURVES OF M



$M = .831$ $CL = .000$ $DX = -.000$ $DY = .021$ $T/C = .110$

FIGURE 6. SYMMETRIC AIRFOIL WITH TWO SUPERSONIC ZONES



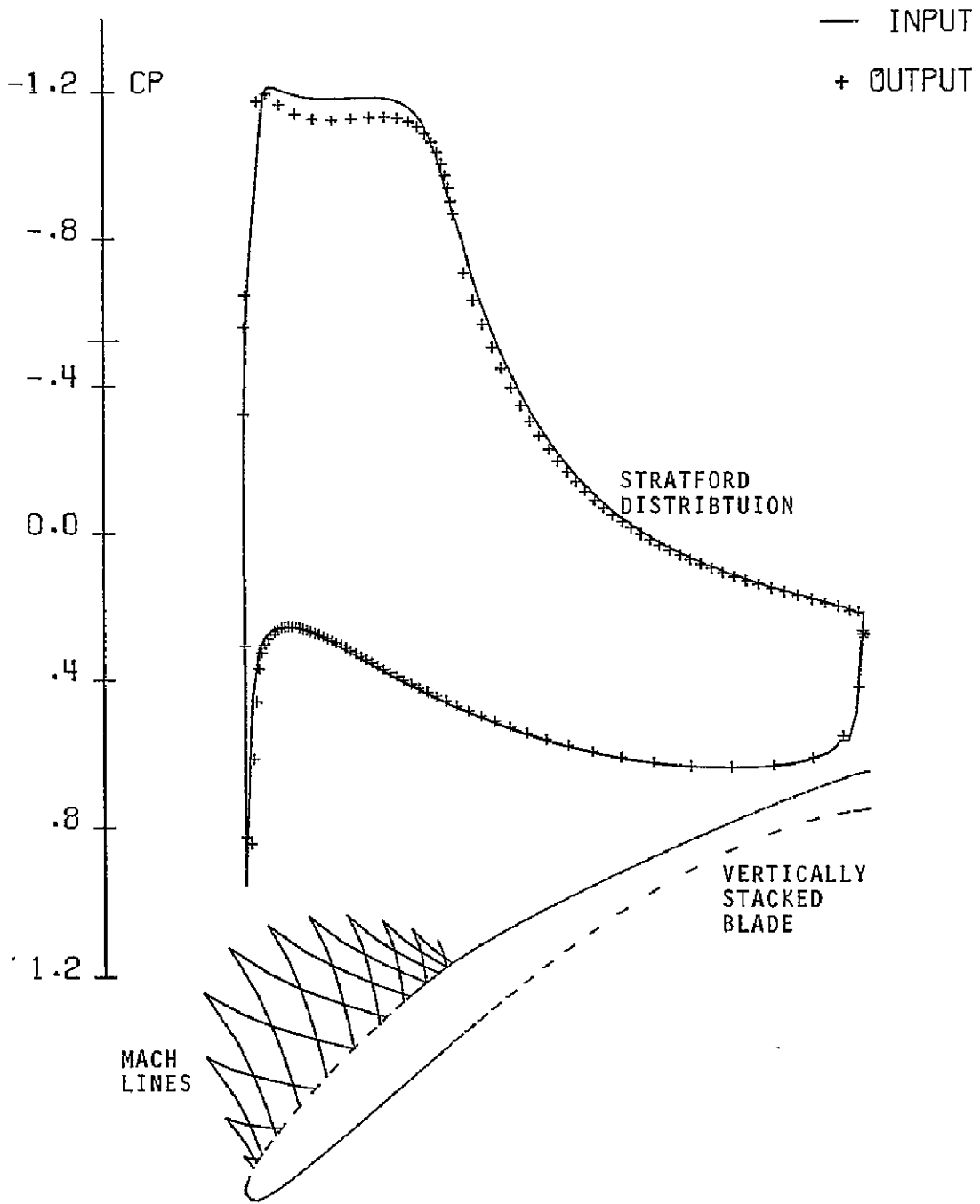
M=.831 CL= .000 DX=-.000 DY= .021 T/C=.110

FIGURE 7. PATHS OF INTEGRATION FOR SYMMETRIC AIRFOIL

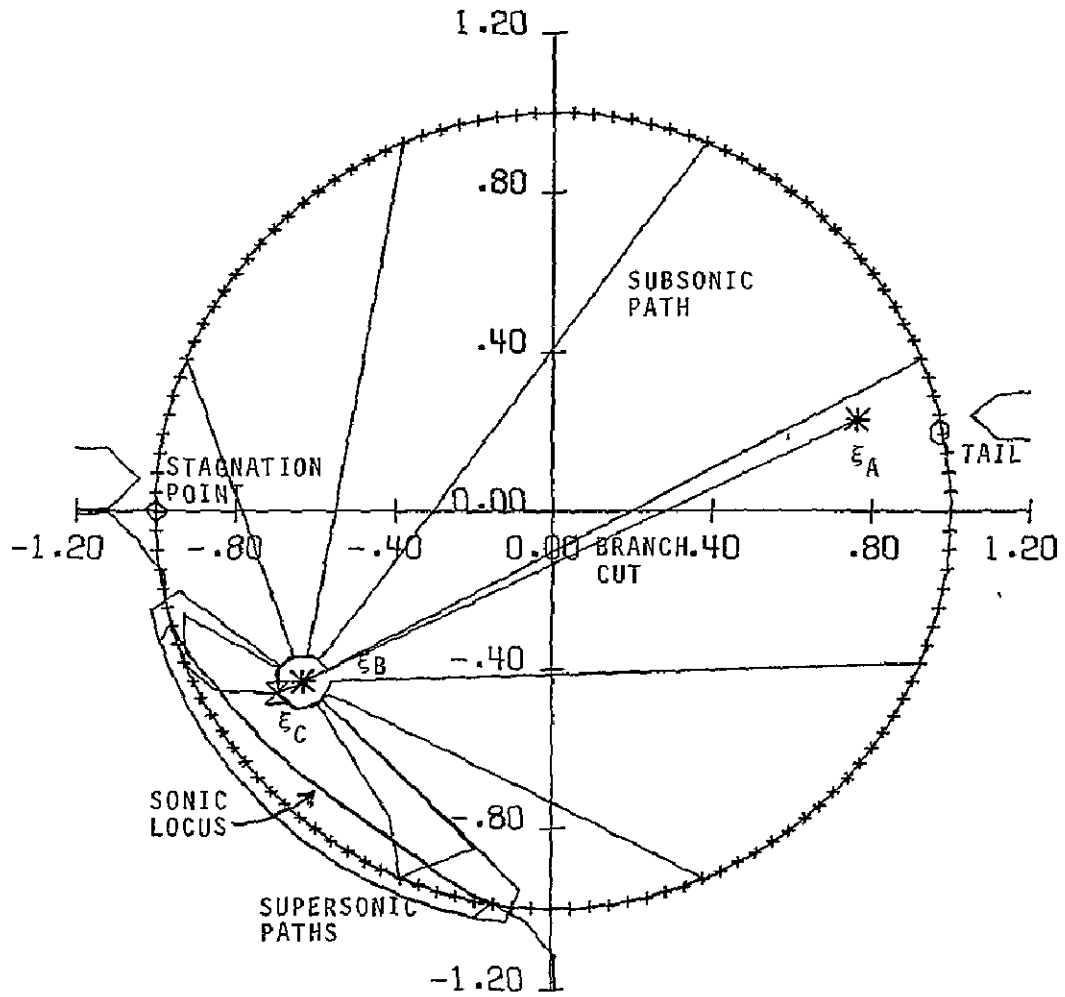
B. Figures 8-13

The input speed distribution for the compressor cascade shown in Figures 8 and 9 was taken from an example prepared, using an earlier version of the design code [23], for Pratt and Whitney Aircraft and tested by the DFVLR in Germany (see also Figure 31). Observe that the Fourier series for the map function fails to approximate the infinite gradient accurately on the lower surface at the trailing edge. Thus airfoils with such extreme aft loading are not always reproduced successfully using the present code.

The objective in the case of Figures 10-13 was to design a compressor cascade with relatively little aft loading and to achieve a monotonically decreasing distribution of thickness near the trailing edge. We arrived at a tail thickness of about 2% after the boundary layer subtraction, and the trailing edge is perpendicular to the upper and lower surfaces. Note that a Stratford pressure distribution has been used, so that SEP is very nearly constant along the upper surface near the trailing edge. The four figures illustrate the complete plot output of a typical run of the design code. For this run the printed output has been listed following the figures. It can serve as a test case for users interested in checking out the code.

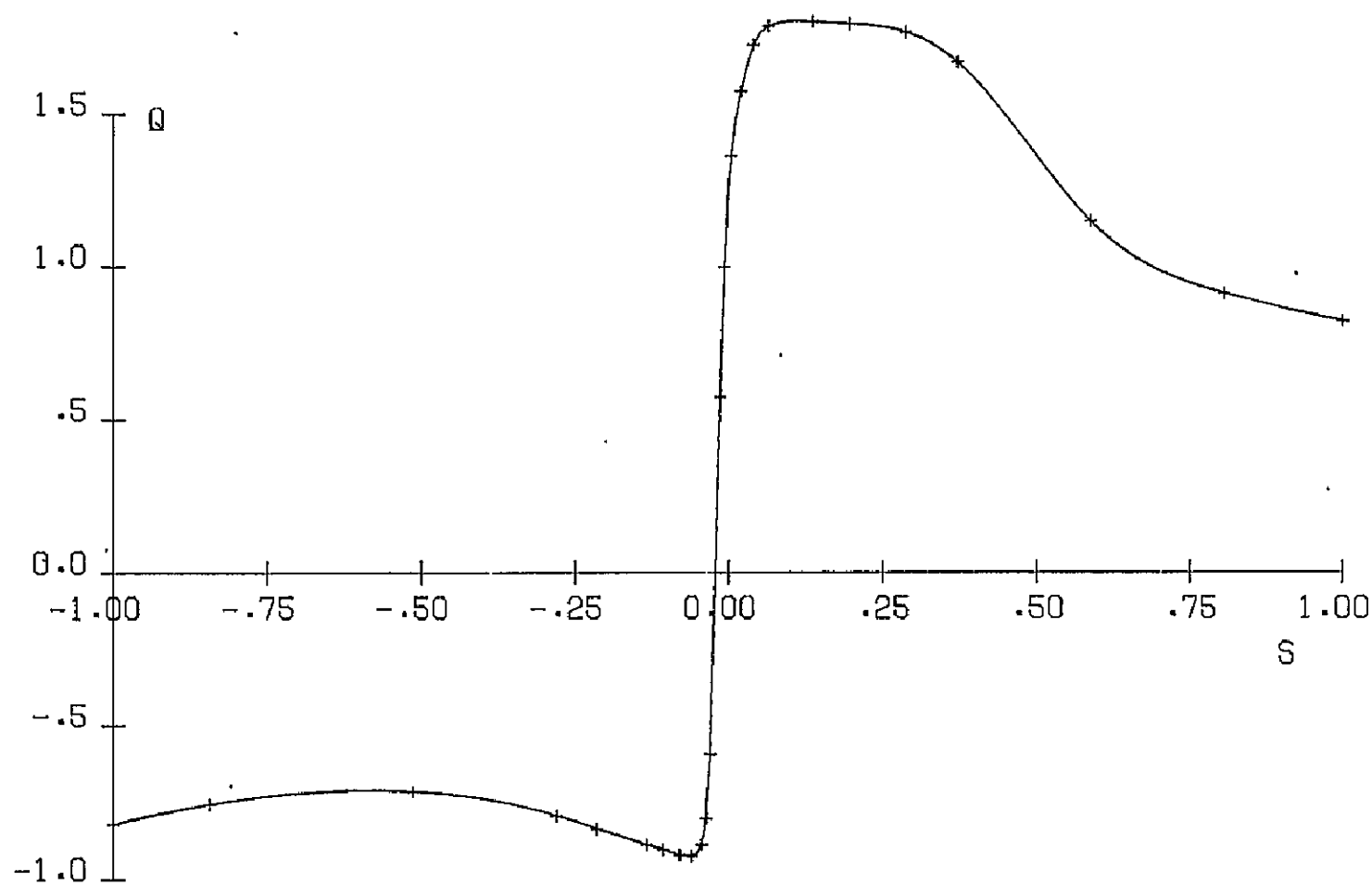


$M_1 = .771$ $M_2 = .484$ $\Delta TH = 24.58$ $G/C = 1.20$
 FIGURE 8. COMPRESSOR AIRFOIL BASED ON PRATT AND WHITNEY EXAMPLE



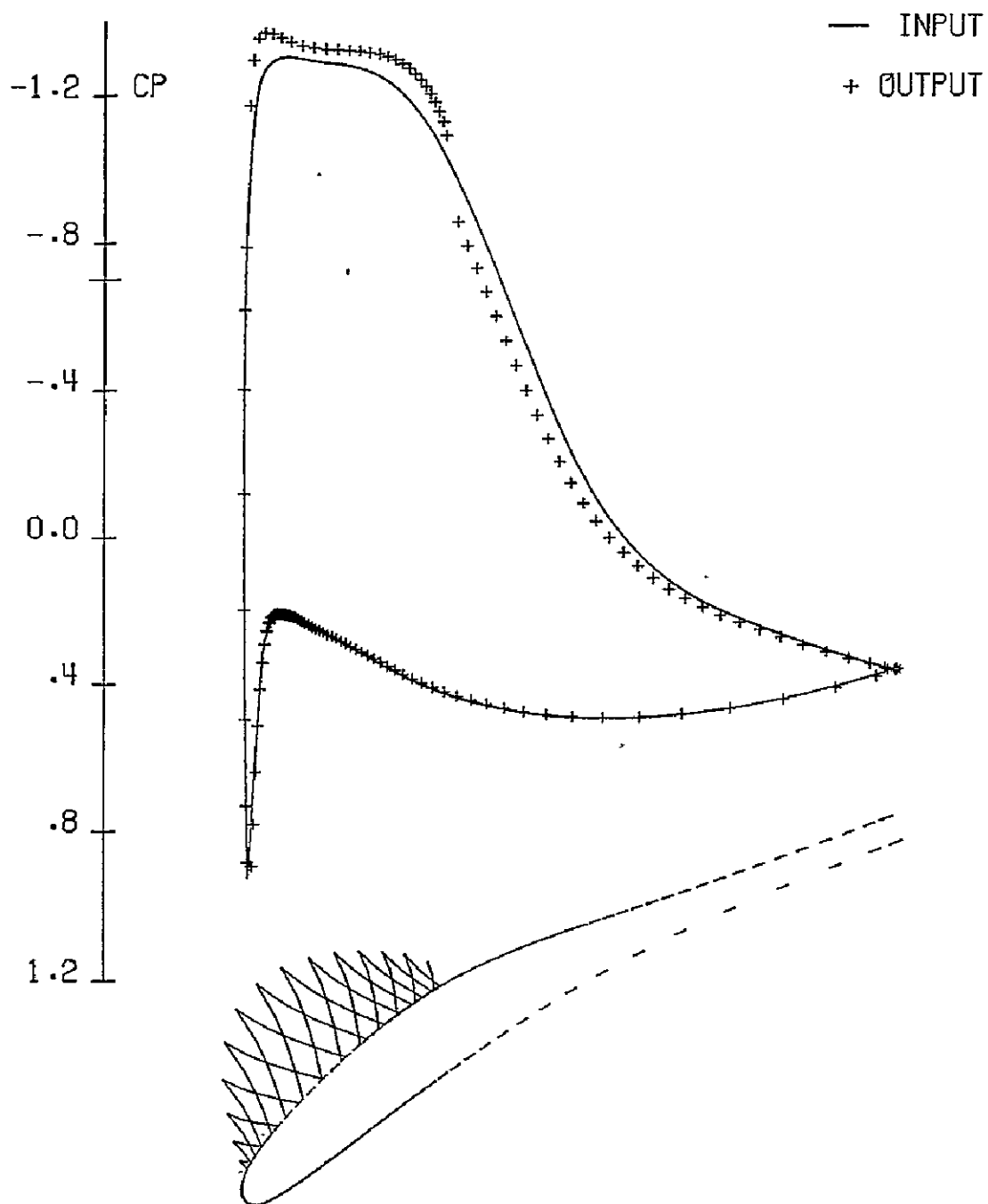
$M_1 = .771$ $M_2 = .484$ $\Delta \theta = 24.58$ $G/C = 1.20$

FIGURE 9. HODOGRAPH PLANE FOR PRATT AND WHITNEY CASCADE



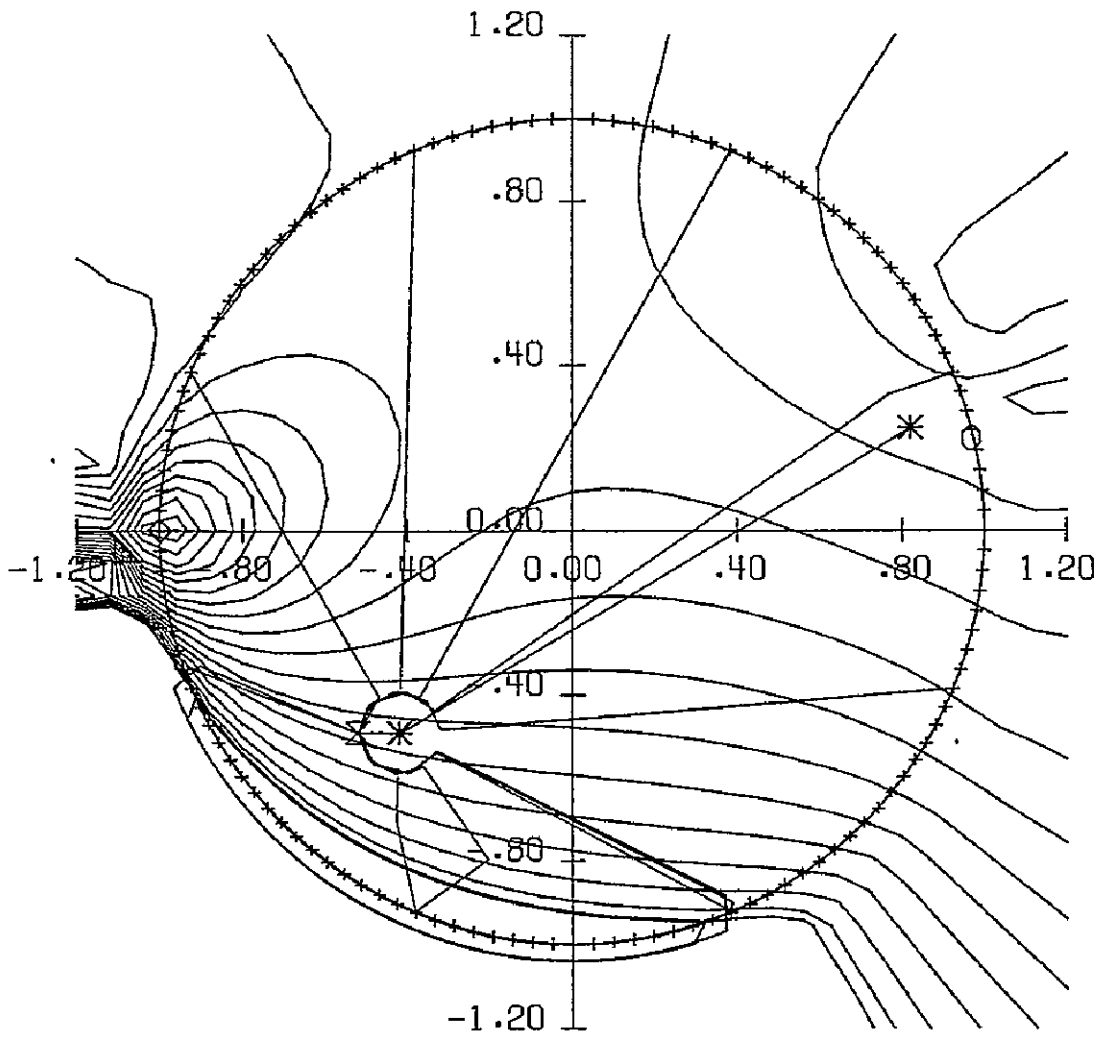
INPUT SPEED DISTRIBUTION

FIGURE 10. INPUT $q(s)$ FOR CASCADE TEST CASE



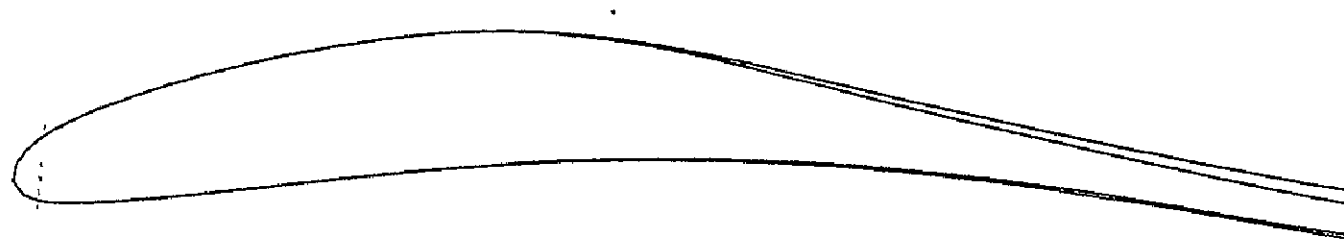
$M_1 = .720$ $M_2 = .484$ $\Delta TH = 21.59$ $G/C = 1.20$

FIGURE 11. TEST CASE CASCADE AIRFOIL



$M_1 = .720$ $M_2 = .484$ $\Delta \theta = 21.59$ $G/C = 1.20$

FIGURE 12. HODOGRAPH PLANE FOR TEST CASE



89

M1=.720 M2=.484 DEL TH= 21.59 G/C=1.20 RN = 1.0 MILLION

FIGURE 13. LARGE PLOT OF COMPRESSOR AIRFOIL

BEGIN EXECUTION OF RUN-339

TAPE3 = INPUT

CARD	S-INPUT	Q-INPUT	S-USED	Q-USED
1	-.010000	-.821000	-1.000000	-.821000
2	.155222	-.756992	-.843941	-.756992
3	.504488	-.717715	-.514044	-.717715
4	.751306	-.795804	-.280913	-.795804
5	.820327	-.838132	-.215719	-.838132
6	.907617	-.889813	-.133271	-.889813
7	.936237	-.905578	-.106238	-.905578
8	.964321	-.922484	-.079711	-.922484
9	.985000	-.925500	-.060179	-.925500
10	1.002000	-.890000	-.044122	-.890000
11	1.010010	-.804954	-.036556	-.804954
12	1.017042	-.595487	-.029914	-.595487
13	1.026123	0.000000	-.021337	0.000000
14	1.034320	.572142	-.013595	.572142
15	1.041325	.996473	-.006978	.996473
16	1.052891	1.361805	.003947	1.361805
17	1.069407	1.571583	.019547	1.571583
18	1.090457	1.722799	.039430	1.722799
19	1.115949	1.782170	.063508	1.782170
20	1.191710	1.797027	.135068	1.797027
21	1.256062	1.790071	.195851	1.790071
22	1.352135	1.763664	.286596	1.763664
23	1.441242	1.665854	.370761	1.665854
24	1.669493	1.146000	.586355	1.146000
25	1.902314	.911000	.806264	.911000
26	2.107425	.821000	1.000000	.821000

TRANSONIC CASCADE DESIGN RUN 339

77/10/03.

TAPE7=INPUT

```

      RUN =-339           ;           MRP = -1
      MACH = .600         ;           RN = 1.0E6
      TRANU = .25         ;           TRANL = .30
      KONE = .30          ;           KTWO = .50
      KTHR = 1.00         ;           NPTS = 201
      NF = 128            ;           NFC = 64
      GAMMA = 1.40        ;           GRID = .08
      NI = 3              ;           NP = 8
      PLTSZ = -8.00       ;           IPLT = 87
XIA = .820 , .250        ;           XIB = -.420 , -.490
      XIC = -.520 , -.490

```

CYCLE	MINLET	MEXIT	RESIDUAL	DX	DY
1	.696	.475	-.170E+00	.05390	.06203
2	.715	.482	-.674E-01	.00893	.03805
3	.720	.484	-.268E-01	-.01243	.03214

LONGEST AUTOMATION PATH HAS 121 POINTS

AUTOMATION CP TIME IS 821.5 SECONDS

LONGEST SUPERSONIC PATH HAS 195 POINTS

SUPERSONIC CP TIME IS 43.4 SECONDS

THICKNESS/CHORD= .1018;

COEFFICIENT OF LIFT=1.3072

DIFFUSION FACTOR= .570

GAP/CHORD= 1.201

INLET MACH NUMBER= .720

INLET FLOW ANGLE= 44.08

EXIT MACH NUMBER= .484

EXIT FLOW ANGLE= 65.67

TURNING ANGLE= 21.59

DX=-.0124

DY= .0321

COORDINATES FROM LOWER SURFACE TAIL TO UPPER SURFACE TAIL

X	Y	ANG	K	M	THETA	SEP	XS	YS
.8718	.4874	21.7	-.16	.4860	.00138	-.00085	.8710	.4892
.8698	.4866	21.8	.17	.4846	.00138	-.00085	.8691	.4884
.8441	.4764	21.5	-.02	.4760	.00139	-.00080	.8434	.4783
.7899	.4551	21.6	-.17	.4616	.00139	-.00066	.7891	.4569
.7196	.4268	22.3	-.30	.4461	.00137	-.00054	.7188	.4287
.6475	.3963	23.7	-.40	.4336	.00132	-.00040	.6467	.3981
.5818	.3664	25.3	-.47	.4257	.00123	-.00024	.5810	.3681
.5246	.3383	27.1	-.51	.4219	.00113	-.00010	.5238	.3398
.4756	.3125	28.7	-.53	.4212	.00101	.00002	.4749	.3138
.4338	.2889	30.1	-.54	.4227	.00090	.00011	.4331	.2901
.3980	.2675	31.4	-.53	.4257	.00079	.00017	.3973	.2686
.3669	.2481	32.5	-.52	.4296	.00068	.00021	.3663	.2490
.3398	.2305	33.5	-.50	.4343	.00058	.00022	.3393	.2313
.3160	.2145	34.3	-.48	.4394	.00049	.00023	.3155	.2151
.2949	.1998	35.1	-.45	.4448			.2949	.1998
.2760	.1864	35.6	-.42	.4504	TRANSITION		.2760	.1864
.2590	.1741	36.1	-.38	.4561			.2590	.1741
.2435	.1628	36.6	-.33	.4618			.2435	.1628
.2295	.1523	36.9	-.28	.4675			.2295	.1523
.2166	.1426	37.2	-.23	.4731			.2166	.1426
.2048	.1336	37.3	-.17	.4786			.2048	.1336
.1938	.1251	37.5	-.11	.4839			.1938	.1251
.1835	.1173	37.6	-.06	.4890			.1835	.1173
.1740	.1099	37.6	-.01	.4938			.1740	.1099
.1650	.1030	37.6	.04	.4984			.1650	.1030
.1565	.0965	37.6	.09	.5026			.1565	.0965
.1485	.0903	37.5	.13	.5066			.1485	.0903
.1409	.0845	37.5	.18	.5103			.1409	.0845
.1337	.0790	37.4	.22	.5138			.1337	.0790
.1268	.0738	37.3	.26	.5171			.1268	.0738
.1203	.0688	37.1	.30	.5202			.1203	.0688
.1140	.0640	37.0	.35	.5230			.1140	.0640
.1079	.0595	36.9	.40	.5257			.1079	.0595
.1021	.0551	36.7	.44	.5282			.1021	.0551
.0965	.0510	36.5	.49	.5306			.0965	.0510
.0911	.0470	36.3	.54	.5328			.0911	.0470
.0859	.0432	36.1	.61	.5350			.0859	.0432
.0808	.0395	35.9	.72	.5373			.0808	.0395

X	Y	ANG	K	M	THETA	SEP	XS	YS
.0760	.0360	35.7	.87	.5396			.0760	.0360
.0712	.0326	35.4	1.08	.5420			.0712	.0326
.0666	.0293	35.0	1.31	.5444			.0666	.0293
.0622	.0262	34.6	1.59	.5467			.0622	.0262
.0579	.0233	34.1	1.87	.5486			.0579	.0233
.0536	.0204	33.6	2.20	.5502			.0536	.0204
.0496	.0178	33.0	2.55	.5514			.0496	.0178
.0456	.0152	32.3	3.03	.5522			.0456	.0152
.0417	.0128	31.5	3.66	.5526			.0417	.0128
.0379	.0105	30.6	4.59	.5528			.0379	.0105
.0343	.0084	29.5	5.86	.5525			.0343	.0084
.0307	.0064	28.1	7.63	.5513			.0307	.0064
.0272	.0047	26.4	9.76	.5482			.0272	.0047
.0238	.0031	24.3	12.58	.5422			.0238	.0031
.0206	.0017	21.7	15.99	.5319			.0206	.0017
.0174	.0005	18.6	21.01	.5160			.0174	.0005
.0143	-.0004	14.7	27.79	.4928			.0143	-.0004
.0113	-.0011	9.8	38.00	.4589			.0113	-.0011
.0084	-.0014	3.4	49.33	.4094			.0084	-.0014
.0055	-.0014	-4.7	62.25	.3389			.0055	-.0014
.0028	-.0010	-14.5	57.44	.2423			.0028	-.0010
0.0000	0.0000	-24.2	62.53	.1261			0.0000	0.0000
-.0050	.0030	-45.2	60.88	.1386			-.0050	.0030
-.0070	.0056	-56.9	51.99	.2798			-.0070	.0056
-.0087	.0087	-67.3	41.52	.4185			-.0087	.0087
-.0099	.0127	-77.1	36.77	.5571			-.0099	.0127
-.0105	.0171	-86.6	27.51	.6877			-.0105	.0171
-.0104	.0225	-95.1	20.81	.8000			-.0104	.0225
-.0096	.0283	-102.1	12.93	.8855			-.0096	.0283
-.0076	.0355	-107.6	11.11	.9539			-.0076	.0355
-.0030	.0471	-115.6	7.77	1.1179			-.0030	.0471
.0017	.0560	-120.0	4.91	1.1745			.0017	.0560
.0082	.0664	-123.5	3.23	1.2020			.0082	.0664
.0164	.0781	-126.1	2.26	1.2096			.0164	.0781
.0263	.0911	-128.2	1.70	1.2080			.0263	.0911
.0378	.1053	-130.0	1.36	1.2033			.0378	.1053
.0511	.1206	-131.6	1.16	1.1982			.0511	.1206
.0658	.1367	-133.1	1.04	1.1941			.0658	.1367
.0815	.1532	-134.4	.97	1.1912			.0815	.1532
.0978	.1694	-135.7	.94	1.1895			.0978	.1694
.1139	.1848	-136.9	.93	1.1885			.1139	.1848
.1294	.1991	-138.0	.95	1.1877			.1294	.1991
.1441	.2120	-139.1	.98	1.1866			.1441	.2120
.1577	.2237	-140.1	1.02	1.1849			.1577	.2237
.1703	.2340	-141.0	1.08	1.1825			.1703	.2340
.1817	.2431	-141.9	1.14	1.1791			.1817	.2431
.1921	.2511	-142.8	1.21	1.1748			.1921	.2511
.2016	.2582	-143.6	1.28	1.1697			.2016	.2582
.2103	.2645	-144.4	1.35	1.1638			.2103	.2645
.2182	.2701	-145.1	1.44	1.1570			.2182	.2701
.2256	.2752	-145.9	1.52	1.1497			.2256	.2752
.2324	.2797	-146.6	1.61	1.1415			.2324	.2797

X	Y	ANG	K	M	THETA	SEP	XS	YS
.2387	.2838	-147.3	1.70	1.1323			.2387	.2838
.2446	.2876	-148.0	1.83	1.1221			.2446	.2876
.2502	.2910	-148.7	1.91	1.1113	TRANSITION		.2502	.2910
.2553	.2941	-149.3	2.02	1.0985			.2553	.2941
.2601	.2969	-150.0	2.14	1.0825	.00026	.00082	.2604	.2964
.2759	.3056	-152.2	1.21	.9832	.00037	.00112	.2762	.3049
.2886	.3122	-153.2	1.35	.9564	.00043	.00075	.2890	.3114
.3009	.3182	-154.2	1.27	.9313	.00049	.00086	.3013	.3174
.3141	.3245	-155.3	1.26	.9054	.00055	.00101	.3145	.3235
.3271	.3303	-156.3	1.16	.8790	.00062	.00118	.3276	.3292
.3409	.3362	-157.3	1.09	.8525	.00070	.00136	.3414	.3350
.3547	.3419	-158.2	.97	.8259	.00079	.00156	.3553	.3404
.3692	.3475	-159.1	.86	.7995	.00089	.00177	.3698	.3459
.3838	.3530	-159.9	.71	.7735	.00101	.00199	.3845	.3511
.3992	.3585	-160.5	.57	.7481	.00114	.00220	.3999	.3564
.4148	.3639	-161.1	.42	.7236	.00128	.00239	.4156	.3615
.4312	.3695	-161.5	.28	.7003	.00144	.00254	.4321	.3667
.4480	.3750	-161.8	.16	.6786	.00161	.00265	.4490	.3719
.4655	.3808	-162.0	.06	.6586	.00180	.00270	.4667	.3773
.4836	.3867	-162.0	-.02	.6402	.00199	.00272	.4849	.3828
.5025	.3928	-162.0	-.08	.6235	.00219	.00269	.5039	.3886
.5221	.3992	-161.9	-.12	.6084	.00240	.00262	.5235	.3947
.5426	.4059	-161.8	-.15	.5947	.00261	.00252	.5442	.4011
.5638	.4130	-161.6	-.16	.5823	.00282	.00239	.5655	.4079
.5862	.4205	-161.3	-.16	.5713	.00303	.00225	.5880	.4151
.6095	.4284	-161.1	-.14	.5612	.00324	.00214	.6114	.4228
.6340	.4368	-160.9	-.13	.5519	.00344	.00208	.6361	.4310
.6598	.4458	-160.7	-.12	.5430	.00366	.00209	.6619	.4396
.6870	.4554	-160.5	-.13	.5341	.00390	.00215	.6894	.4488
.7156	.4656	-160.3	-.15	.5251	.00415	.00222	.7182	.4585
.7458	.4765	-160.0	-.18	.5162	.00443	.00228	.7485	.4689
.7768	.4878	-159.7	-.23	.5074	.00472	.00232	.7797	.4798
.8077	.4994	-159.2	-.29	.4993	.00501	.00231	.8109	.4910
.8357	.5102	-158.7	-.37	.4925	.00528	.00223	.8391	.5014
.8554	.5180	-158.3	-.12	.4882	.00546	.00219	.8590	.5089
.8593	.5195	-158.3	0.00	.4860	.00549	.00219	.8629	.5104

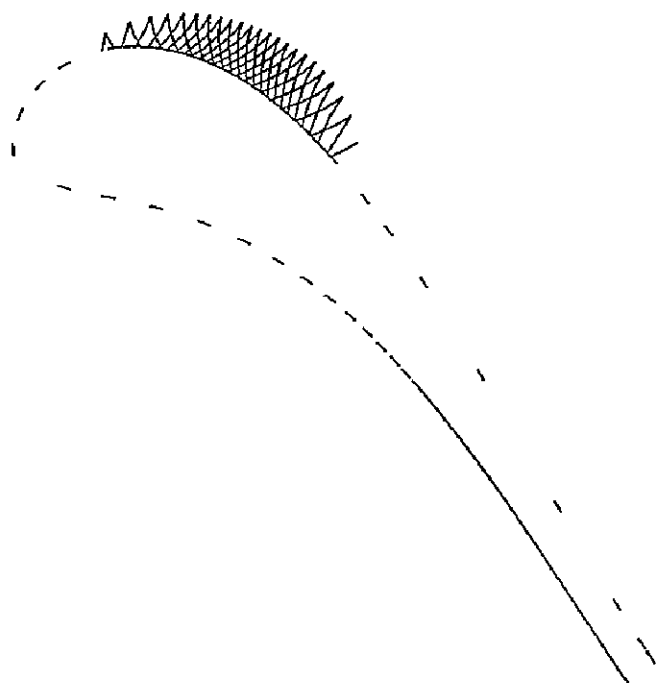
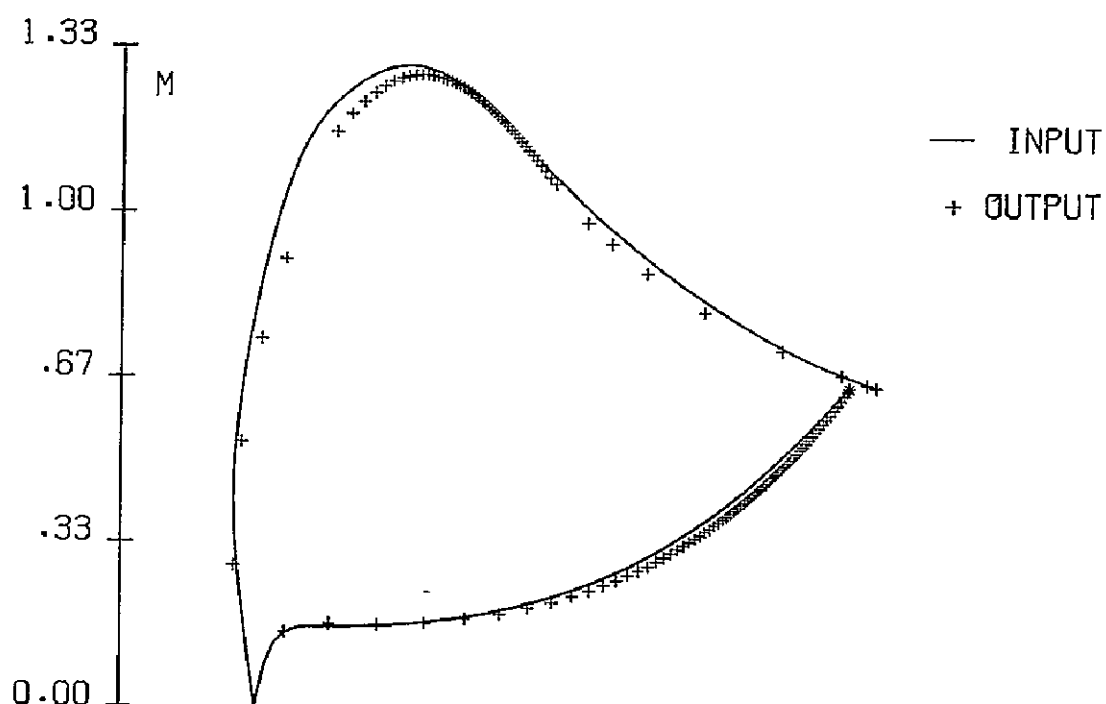
PROFILE DRAG COEFFICIENT = .0093

LOSS COEFFICIENT = .0166

C. Figures 14-18

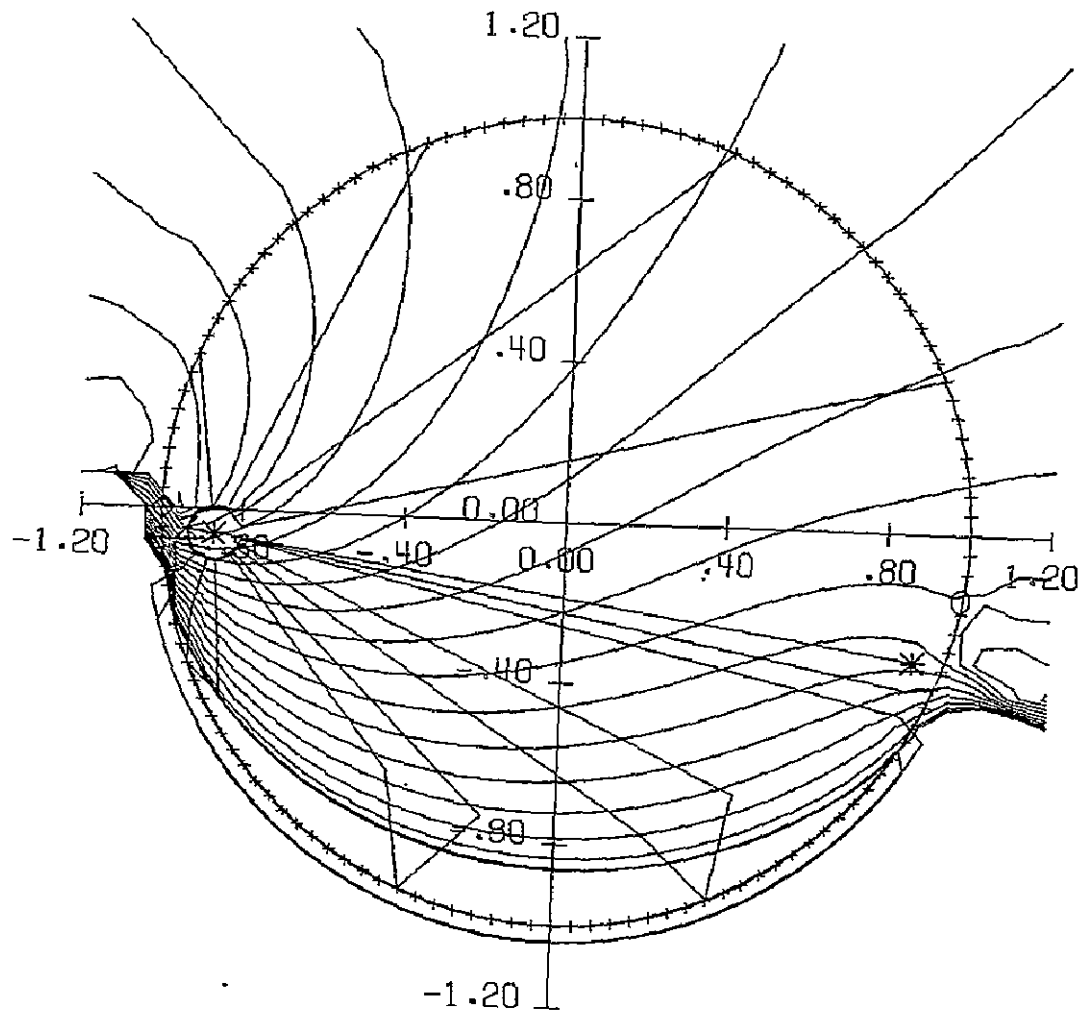
In Figures 14 and 15 we give an example of a transonic turbine cascade that follows work of McIntyre [24]. The attempt is somewhat academic, but the calculated airfoil is perhaps of interest as a first vane. Major difficulties were encountered because of wide separation between ξ_A and ξ_B , which is the result of high solidity and the difference between M_1 and M_2 . Unfortunately the supersonic zone cannot be located very far back on the airfoil because that makes the supersonic paths of integration cross branch cuts in a way that is not permitted by the present version of the design code.

In Figures 16-18 we present a more dramatic turbine cascade that might serve as a root section. The turning angle is 118° , but the supersonic zone is very slender for the reasons just described. It is possible to modify the code to overcome these difficulties (see Section 3 of Chapter III and Section 2 of Chapter VI).



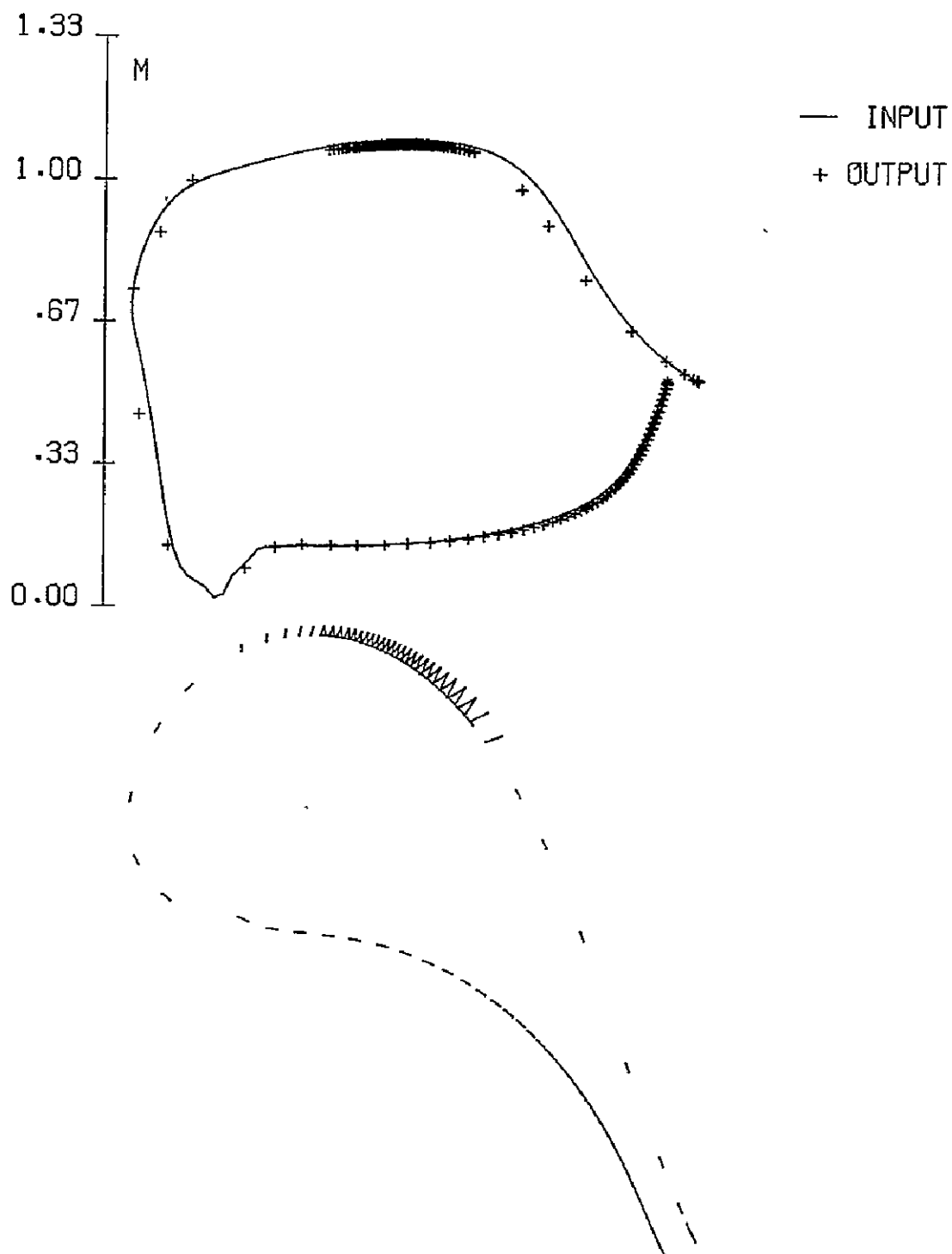
$M_1=.325$ $M_2=.732$ $\Delta T_H=74.28$ $G/C=1.06$

FIGURE 14. TURBINE AIRFOIL BASED ON MCINTYRE CASCADE



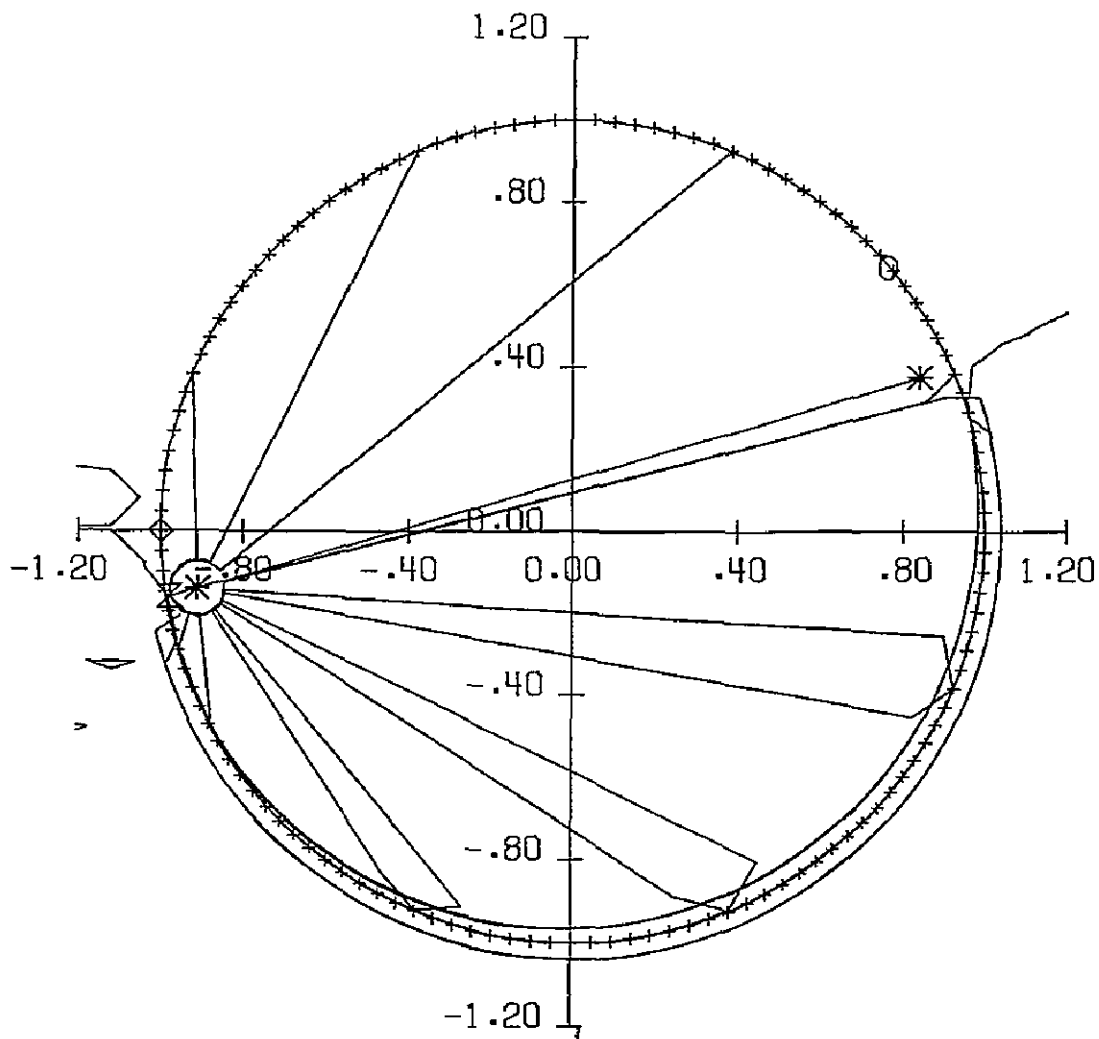
$M_1 = .325$ $M_2 = .732$ $\Delta \theta = 74.28$ $G/C = 1.06$

FIGURE 15. HODOGRAPH PLANE FOR MCINTYRE EXAMPLE



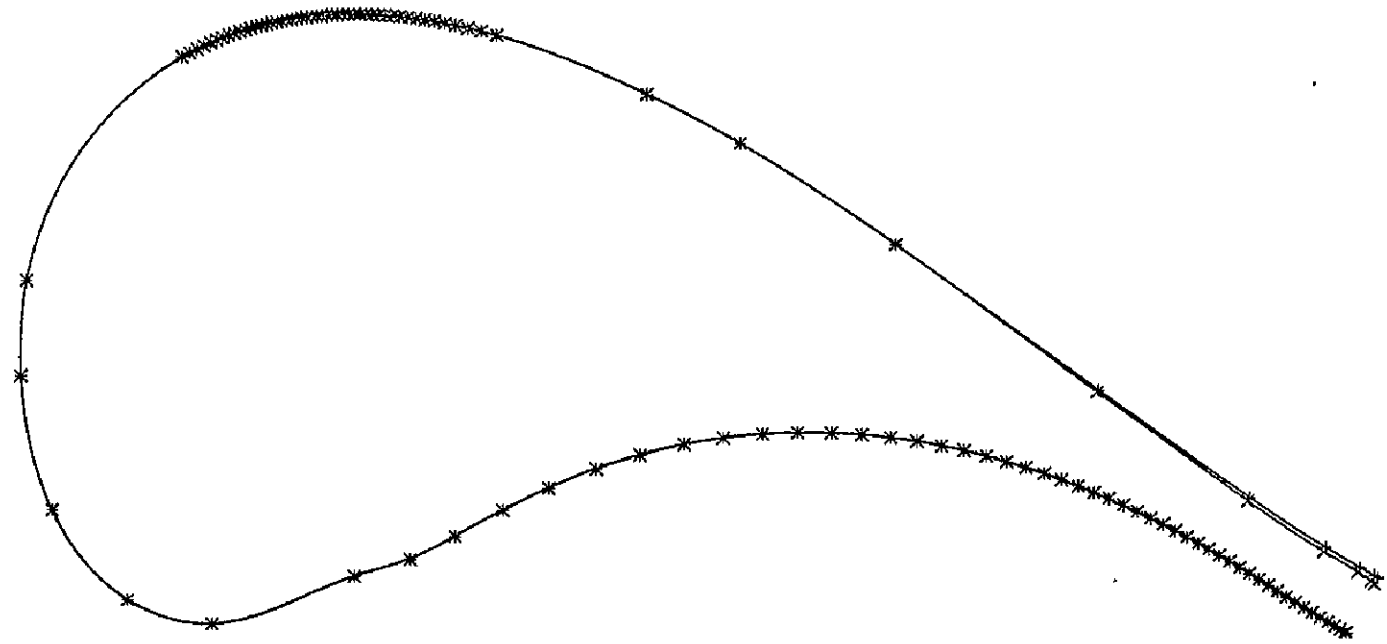
$M_1 = .410$ $M_2 = .667$ $\Delta \theta = 118.28$ $G/C = 1.16$

FIGURE 16. TURBINE CASCADE WITH LARGER TURNING ANGLE



M1=.410 M2=.667 DEL TH=118.28 G/C=1.16

FIGURE 17. PATHS OF INTEGRATION FOR TURBINE CASCADE



71

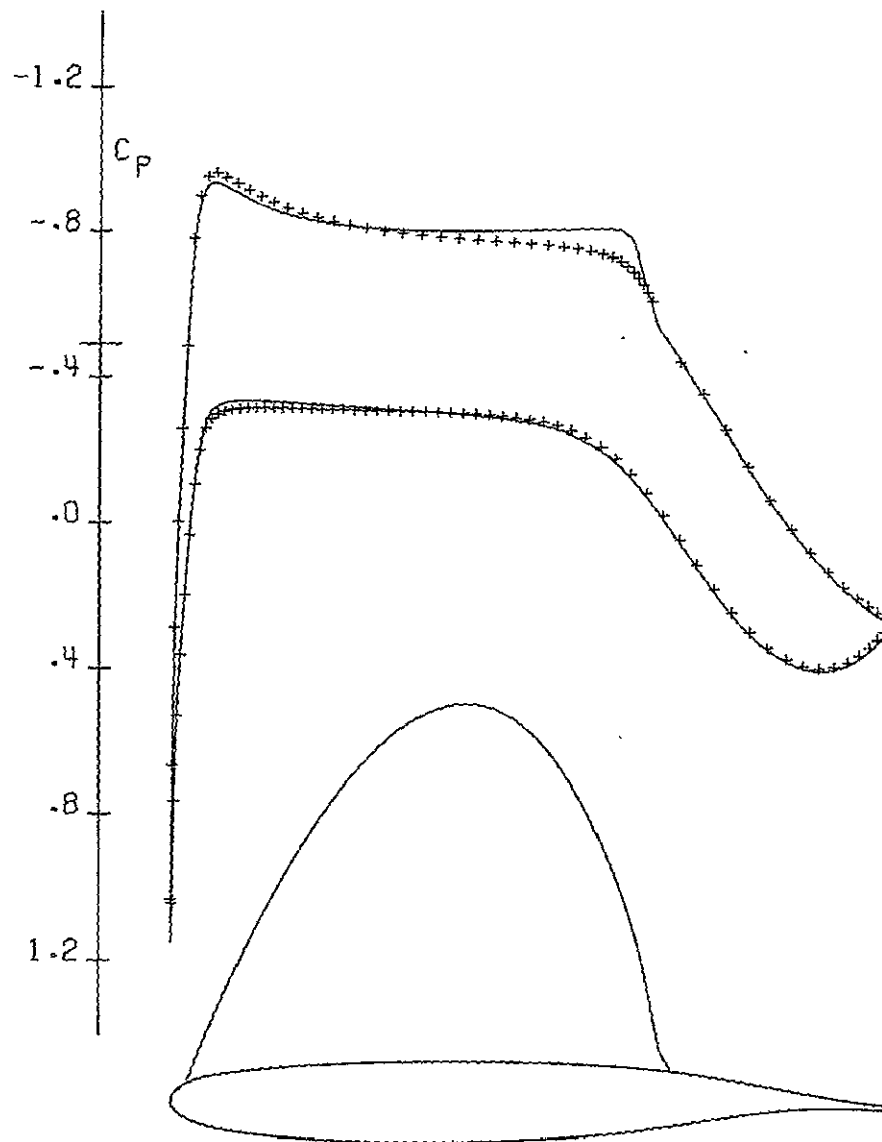
M1=.410 M2=.667 DEL TH=118.28 G/C=1.16 RN = 1.0 MILLION

FIGURE 18. GEOMETRY OF TURBINE CASCADE AIRFOIL

2. Data from Analysis and Experiment

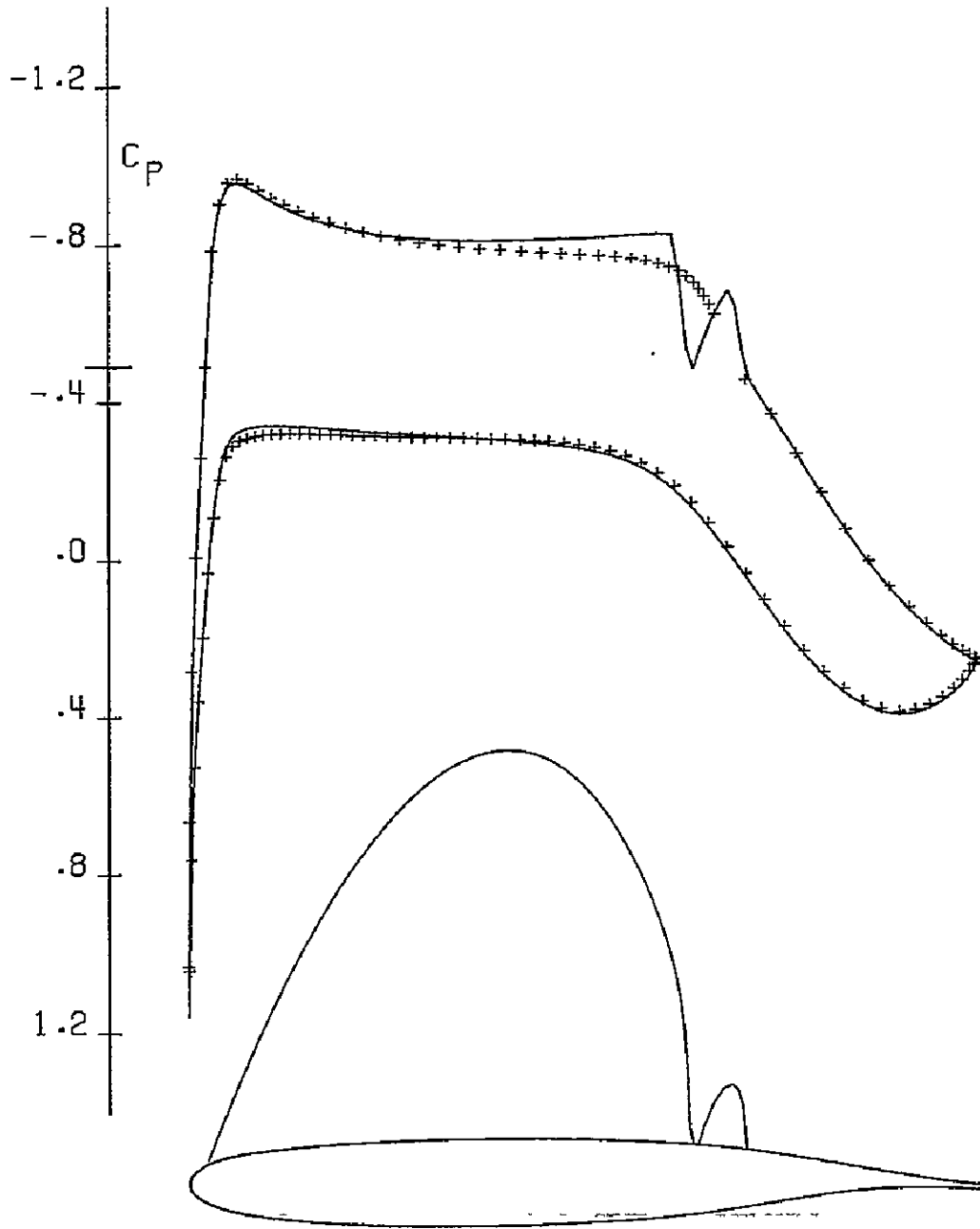
A. Figures 19-21

Runs of the design and analysis codes are compared in Figures 19-21. The airfoil used is that shown in Figure 2. In Figure 19 the results of a design run performed with $MRP = 8$ and $NFC = 64$ are compared with an NCF run of the analysis code on a grid of 160×30 points. The agreement is good enough to give some confidence that both codes have no bugs that might be physically significant. Similar agreement is seen in Figure 20, where the analysis calculation was performed on a mesh of 320×60 points. For this grid lack of uniqueness in the NCF solution led to questionable convergence (see Volume II). Figure 21 shows a comparison between the design and analysis computations when the boundary layer correction is made in both cases. The agreement is not quite as good as before, but does confirm that a Stratford pressure distribution near the trailing edge eliminates any significant loss of lift in passing from design to analysis.



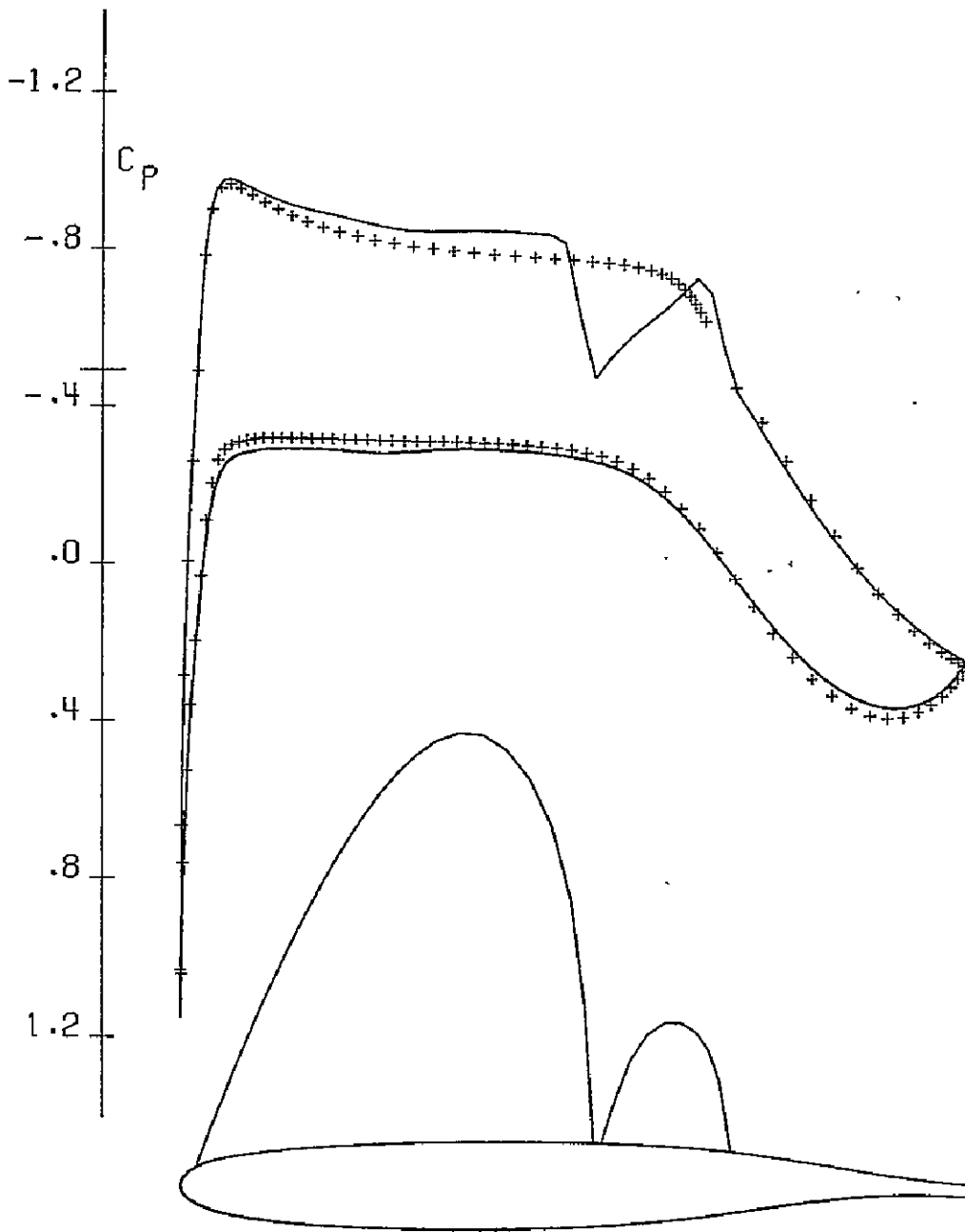
AIRFOIL 78-05-11 M*N=160*30 NCY= 10 NO VISCOSITY
 — ANALYSIS M=.781 ALP= -.11 CL= .479 CD=.0005
 + DESIGN M=.781 ALP= 0.00 CL= .479 CD=.0000

FIGURE 19. COMPARISON OF DESIGN AND ANALYSIS ON STANDARD GRID



AIRFOIL 78-05-11 M*N=320*60 NCY= 5 NO VISCOSITY
 — ANALYSIS M=.781 ALP= -.11 CL= .479 CD=.0006
 + DESIGN M=.781 ALP= 0.00 CL= .479 CD=.0000

FIGURE 20. COMPARISON OF DESIGN AND ANALYSIS ON FINER GRID



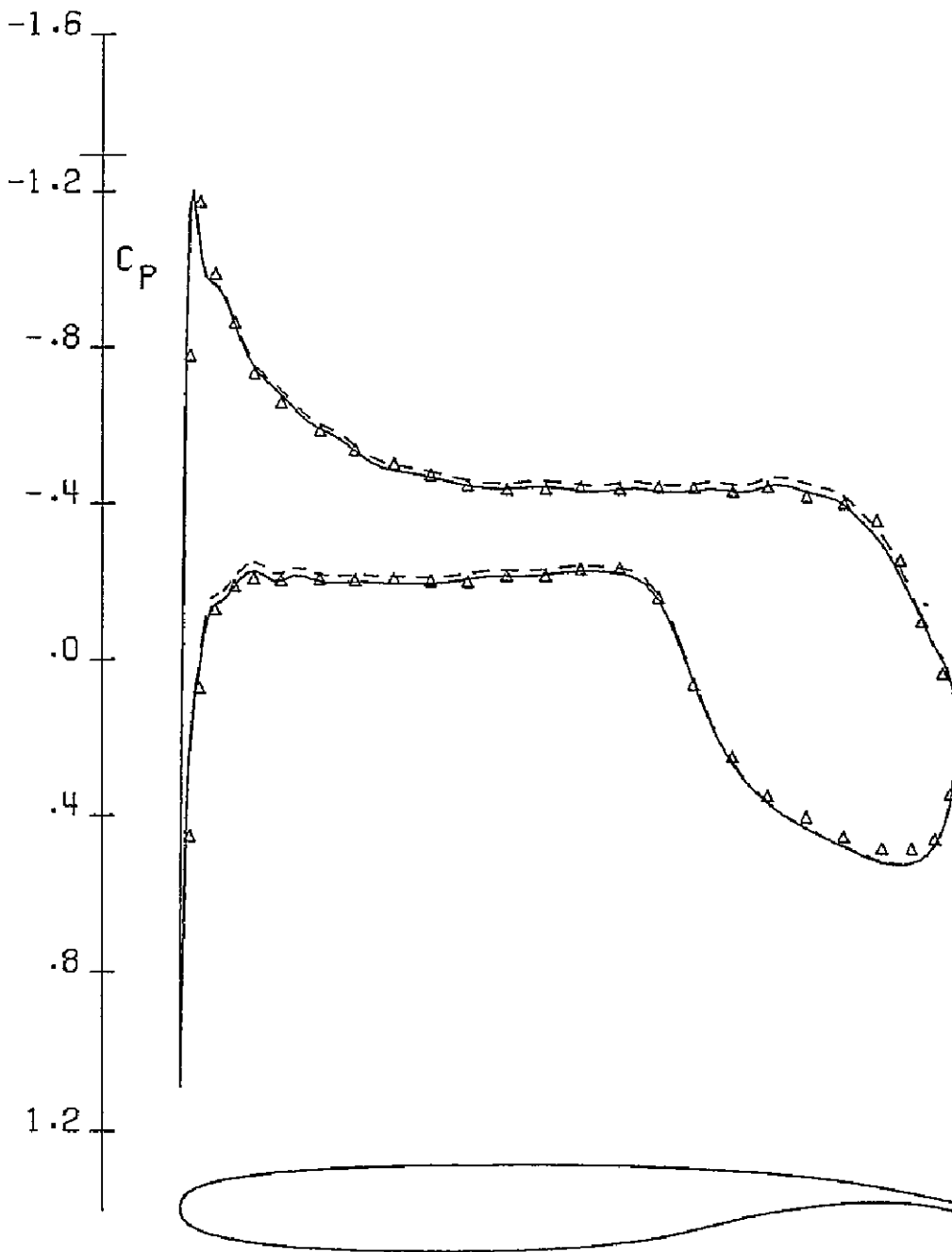
AIRFOIL 78-05-11 M*N=160*30 NCY= 10 R= 7 MILLION
 — THEORY M=.781 ALP= .31 CL= .479 CD=.0085
 + DESIGN M=.781 ALP= 0.00 CL= .479 CD=.0000

FIGURE 21. DESIGN VERSUS ANALYSIS WITH BOUNDARY LAYER CORRECTION

B. Figures 22-25

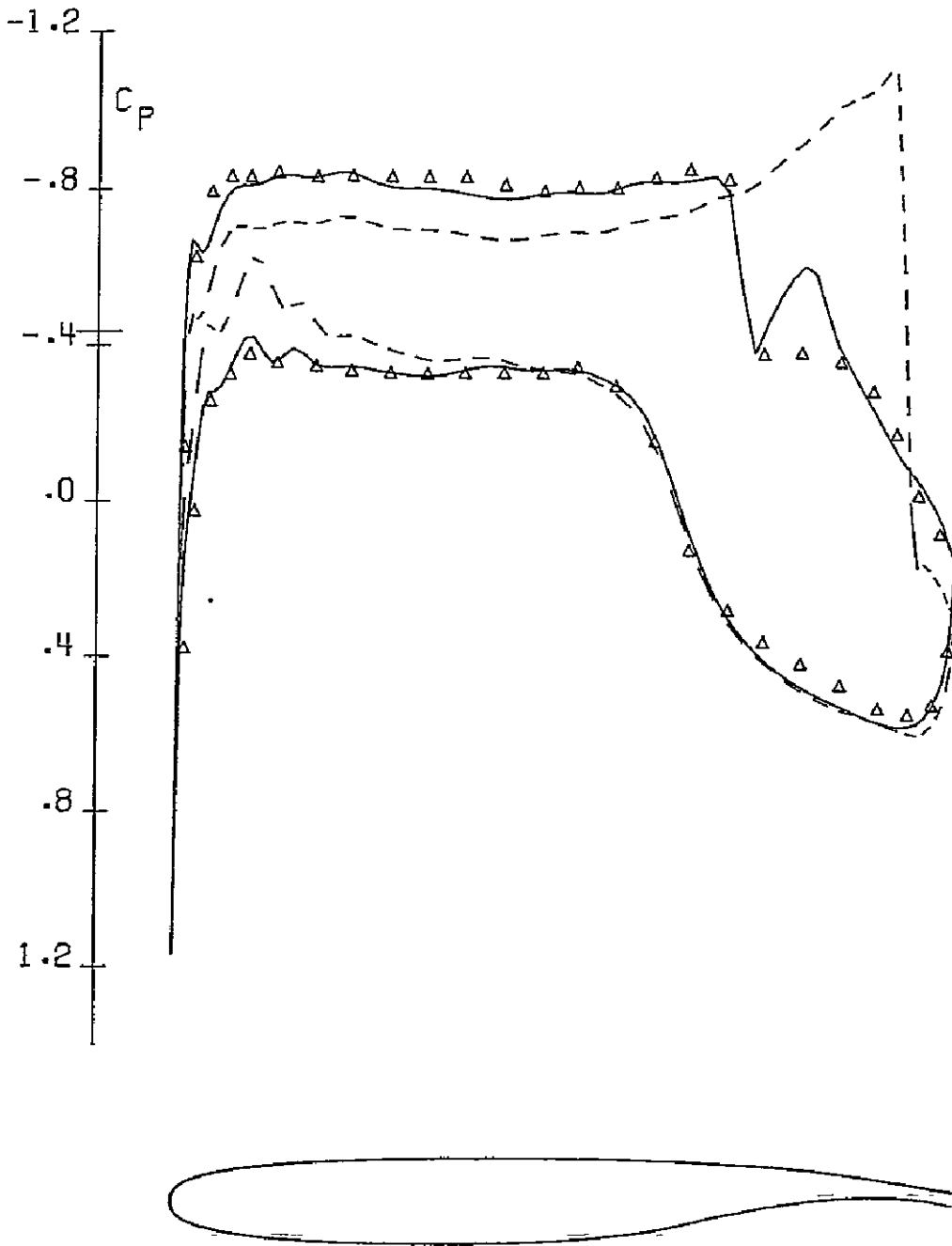
The analysis code is now in use by more than twenty industrial laboratories in the United States. It is based on data obtained from wind tunnel tests for two of the airfoils we designed [2,12,19,20,21]. Additional computations with further airfoils seem to justify the use of semi-empirical procedures. In particular, we have also analyzed the original Whitcomb wing [33]. The results are shown in Figures 22-25, which compare the NCF and FCF methods [2,16].

Agreement between theory and experiment is typically excellent for subsonic cases even when there is heavy aft loading, as illustrated in Figure 22. Figure 24 shows a run for the Whitcomb airfoil where the shock location is not so well predicted by either the NCF or FCF method. However, agreement between test data and our analysis corrected for the boundary layer effect is generally not worse than it is in this case. Figure 25 is a high lift run where agreement with the NCF calculation remains good despite the fact that the flow is separated and nearing buffet.



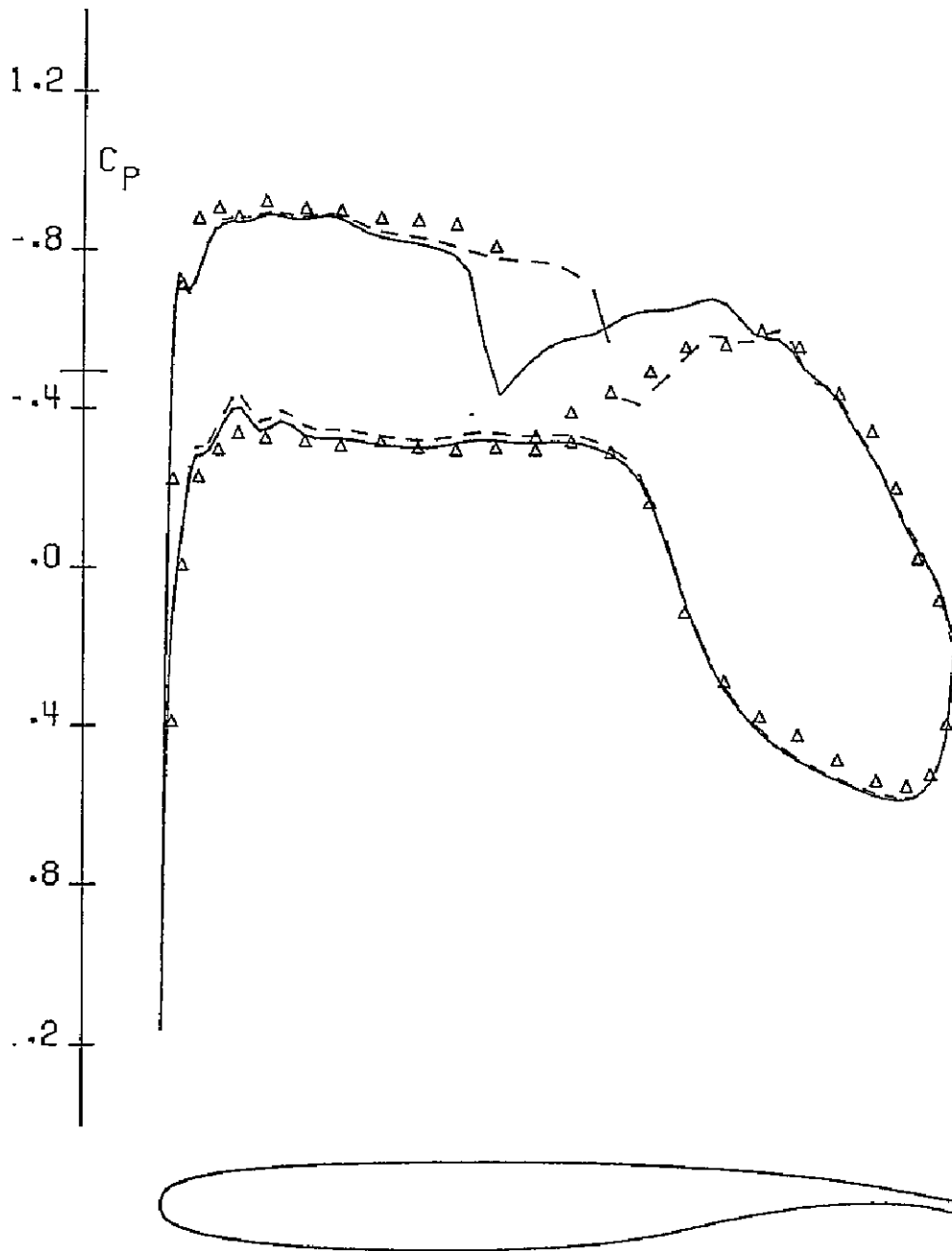
WHITCOMB WING	M*N=160*30	NCY= 10	R= 7 MILLION
— NCF THEORY	M=.600	ALP= .05	CL= .495 CD=.0069
-- FCF THEORY	M=.600	ALP= 0.00	CL= .495 CD=.0074
Δ EXPERIMENT	M=.600	ALP= 1.00	CL= .495 CD=.0082

FIGURE 22. COMPARISON OF ANALYSIS AND EXPERIMENT FOR WHITCOMB WING



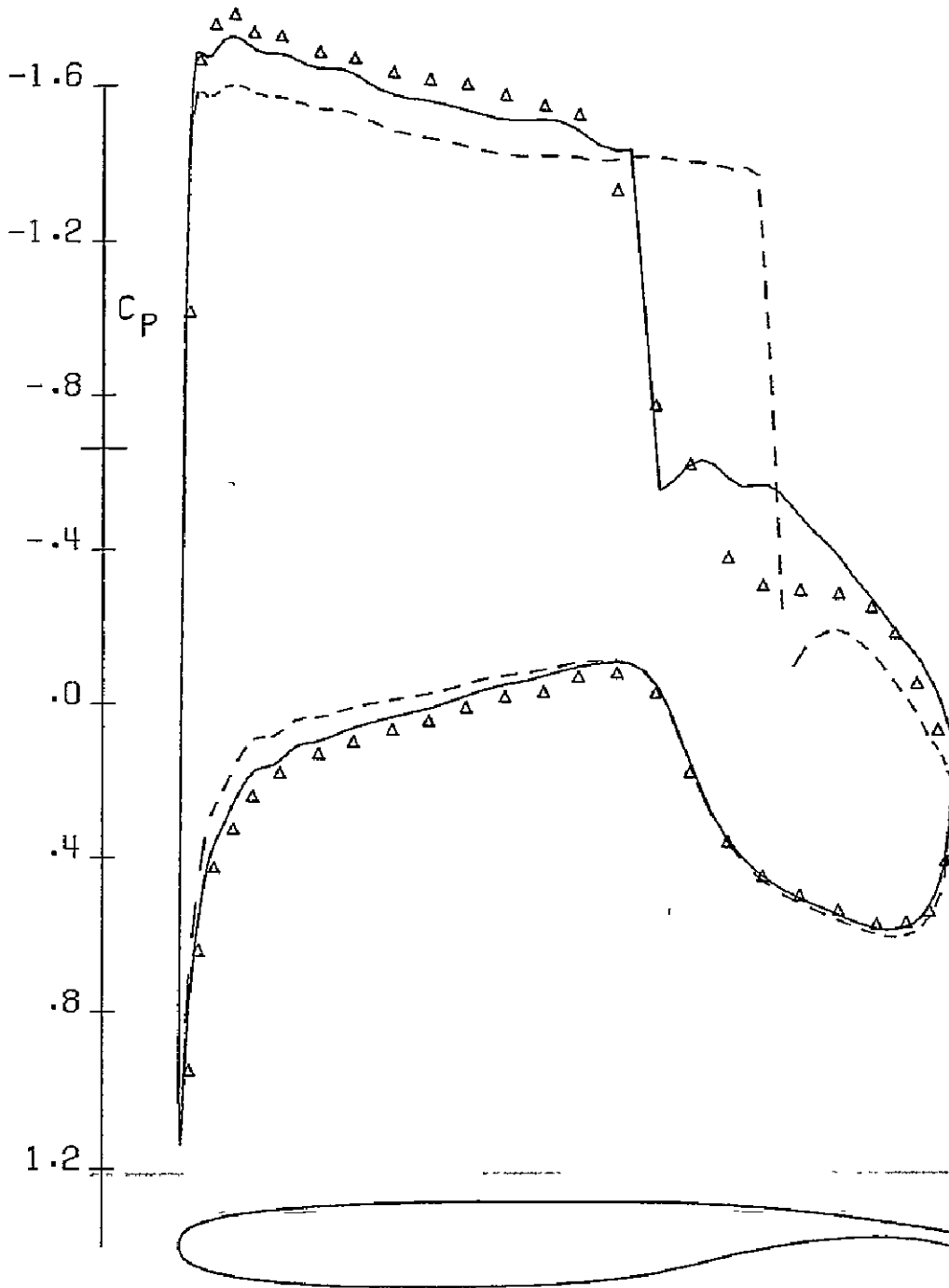
WHITCOMB WING	M*N=160*30	NCY= 10	R= 8 MILLION
— NCF THEORY	M=.800	ALP= -.15	CL= .613 CD=.0111
-- FCF THEORY	M=.800	ALP=-1.40	CL= .613 CD=.0118
Δ EXPERIMENT	M=.800	ALP= 1.00	CL= .613 CD=.0110

FIGURE 23. ANALYSIS VERSUS EXPERIMENT AT THE DESIGN CONDITION



IHITCOMB WING	M*N=160*30	NCY= 10	R= 8 MILLION
— NCF THEORY	M=.780	ALP= -.28	CL= .576 CD=.0089
-- FCF THEORY	M=.780	ALP= -.28	CL= .576 CD=.0107
△ EXPERIMENT	M=.780	ALP= 1.00	CL= .576 CD=.0098

FIGURE 24. OFF-DESIGN CASE WITH TWO SHOCKS

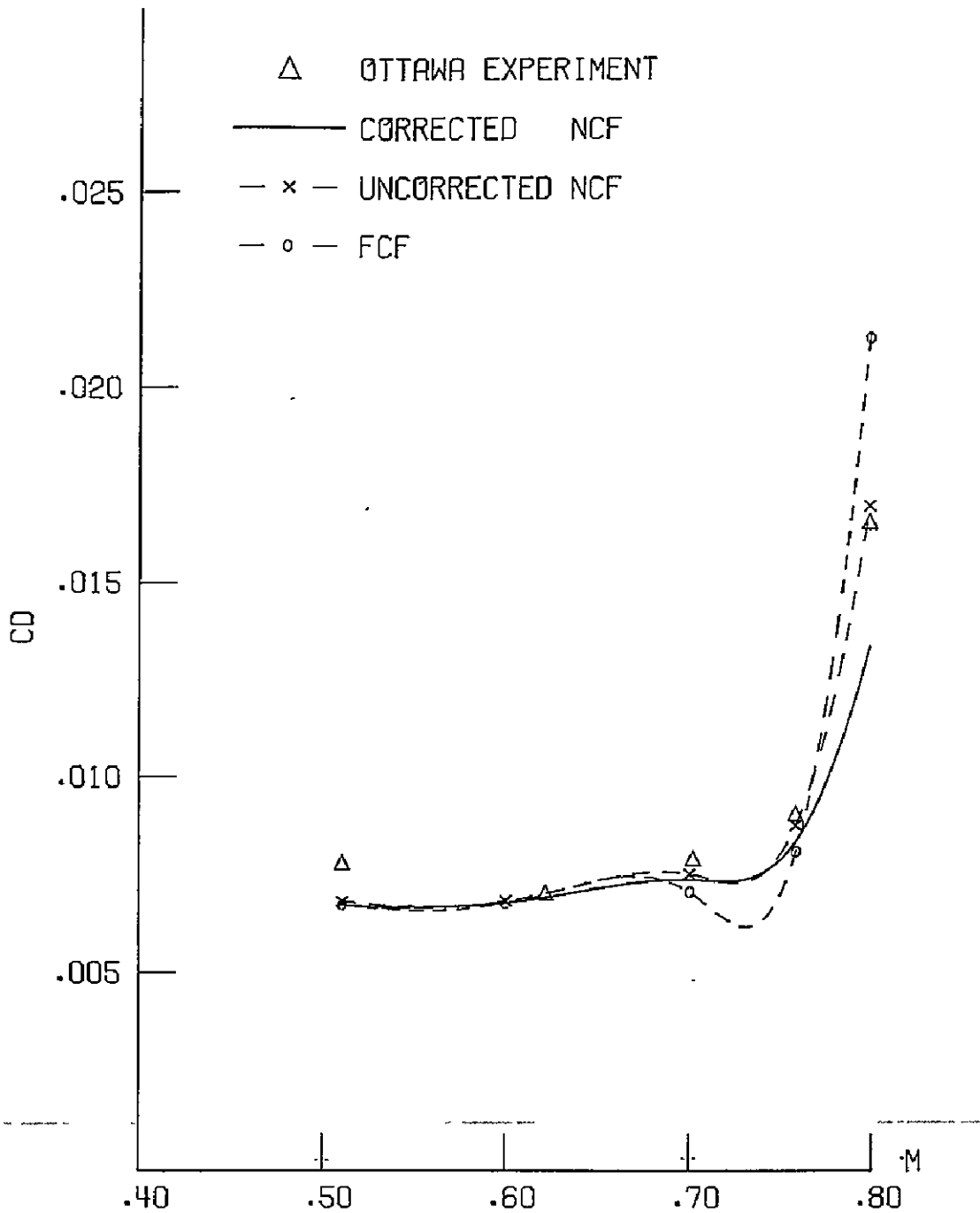


WHITCOMB WING	M=0.730	ALP= 3.57	CL=1.315	CD=.0391
— NCF THEORY	M=0.730	ALP= 1.90	CL=1.315	CD=.0341
-- FCF THEORY	M=0.730	ALP= 5.50	CL=1.315	CD=.0630
△ EXPERIMENT				

FIGURE 25. HIGH LIFT CASE NEAR BUFFET

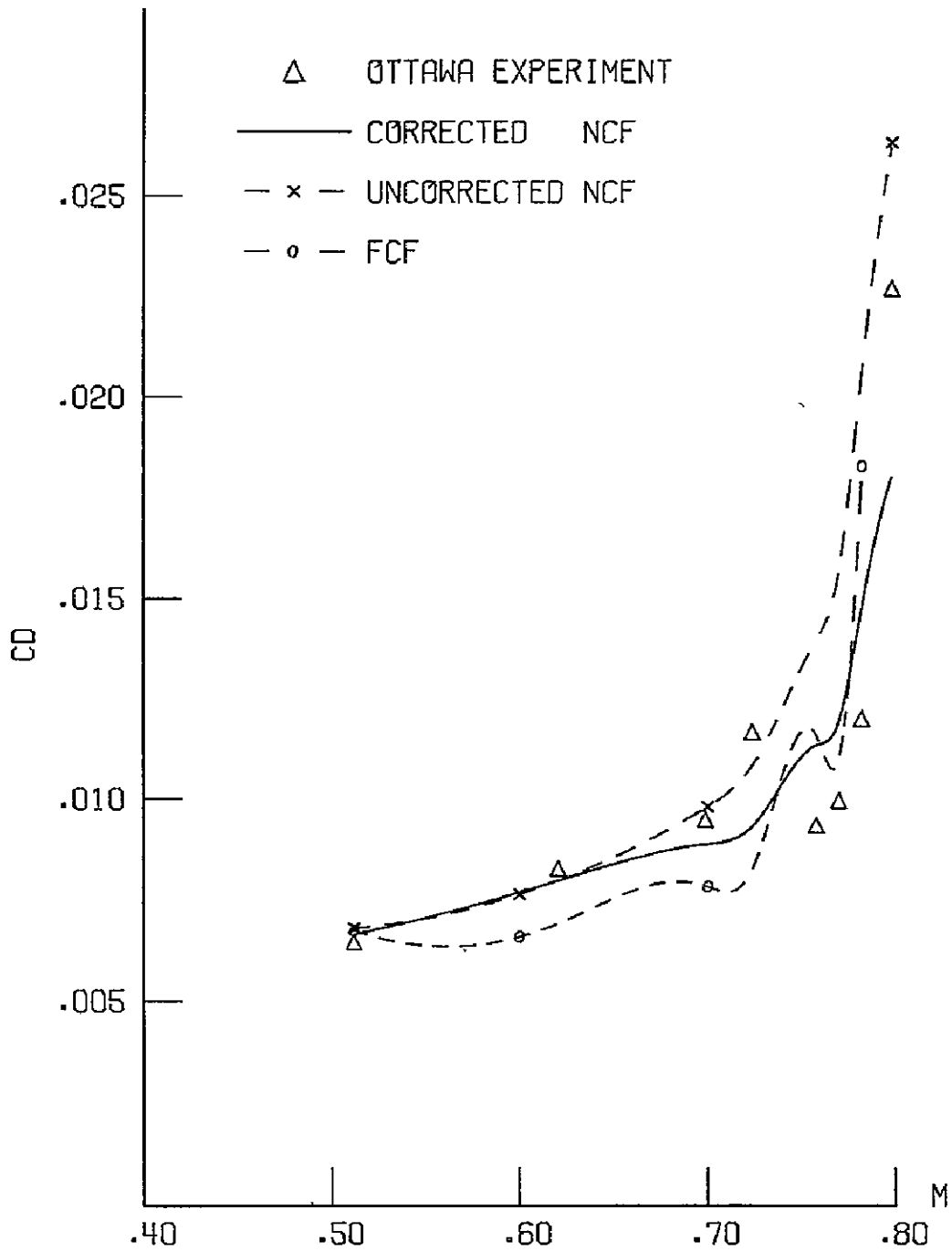
C. Figures 26-29

Figures 26 and 27 show a comparison for airfoil 75-06-12 (see Volume II) of drag rise curves at $C_L = .4$ and at $C_L = .6$ for our theory and the Ottawa wind tunnel tests [19,20,21]. Figures 28 and 29 present a similar comparison for airfoil 75-07-15 (see Volume II) of drag polars at $M = .69$ and at $M = .76$. For the higher lift coefficients and Mach numbers, which are associated with large supersonic zones and shock waves interacting with relatively thick turbulent boundary layer, the corrected NCF method is seen to give the best results (see Section 1 of Chapter IV). In the lower range of C_L and M , which corresponds to shocks nearer the leading edge where the boundary layer is thinner, there is a less decisive difference between the uncorrected NCF, corrected NCF and FCF methods. In general we prefer the corrected NCF results, which usually lie below the uncorrected NCF values but above the FCF values of the drag coefficient. This is the method that has been included in our update of the analysis code H.



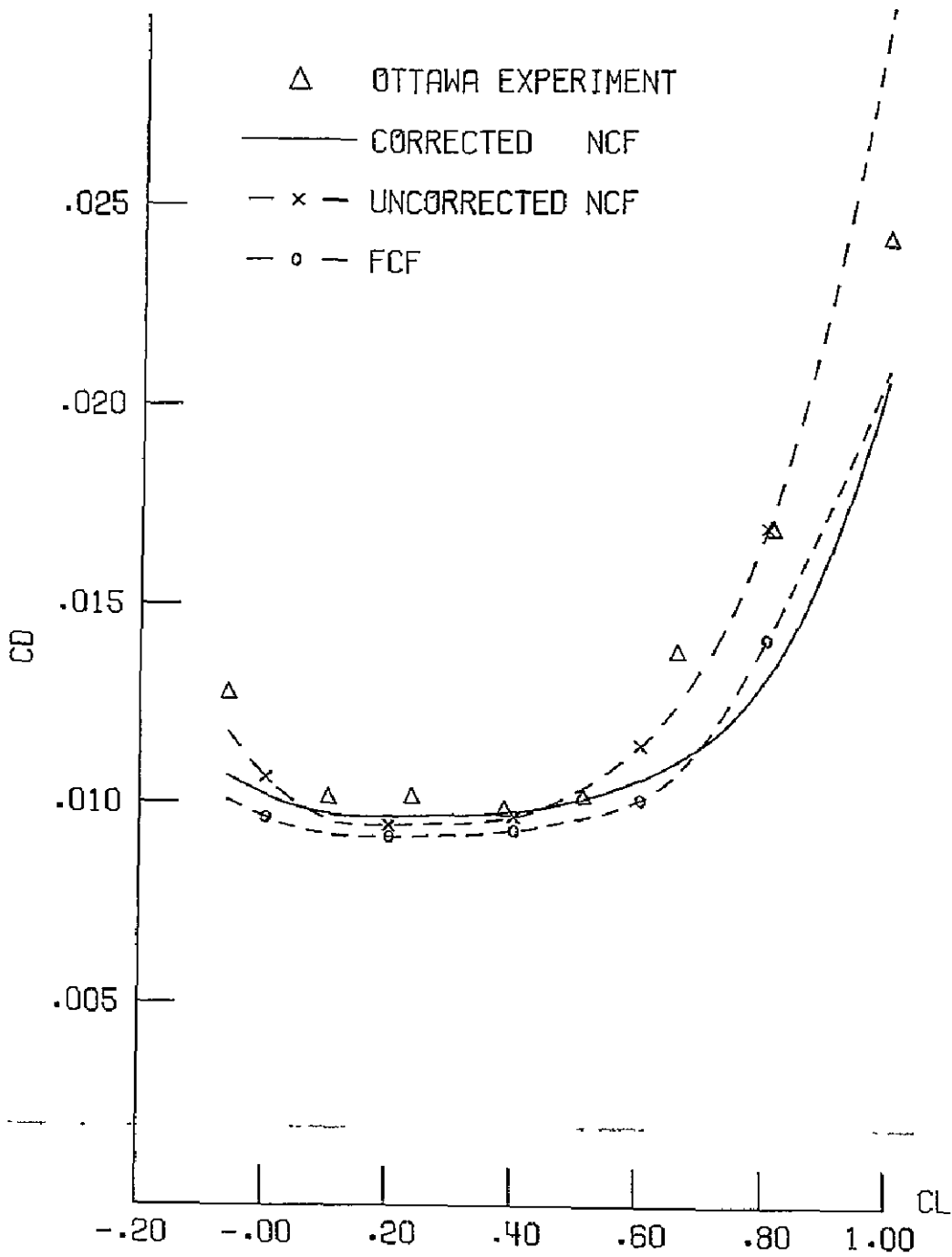
DRAG RISE CURVES FOR AIRFOIL 75-06-12, $CL=0.4$

FIGURE 26. THEORETICAL VERSUS EXPERIMENTAL DRAG RISE CURVES



DRAG RISE CURVES FOR AIRFOIL 75-06-12, $CL=0.6$

FIGURE 27. DRAG RISE CURVES NEAR THE SHOCKLESS CONDITION



DRAG POLARS FOR AIRFOIL 75-07-15 AT $M=0.69$

FIGURE 28. THEORETICAL VERSUS EXPERIMENTAL DRAG POLARS

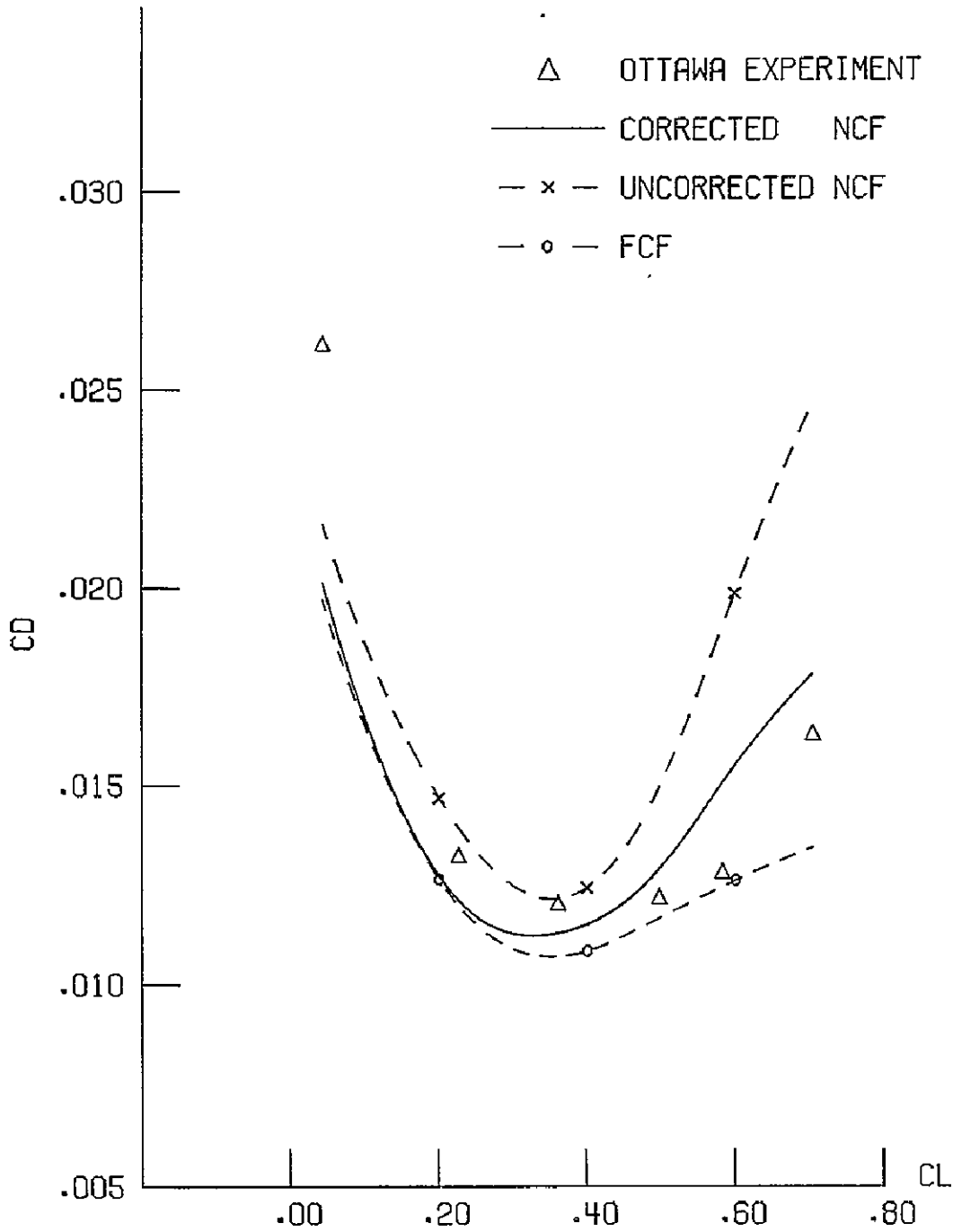
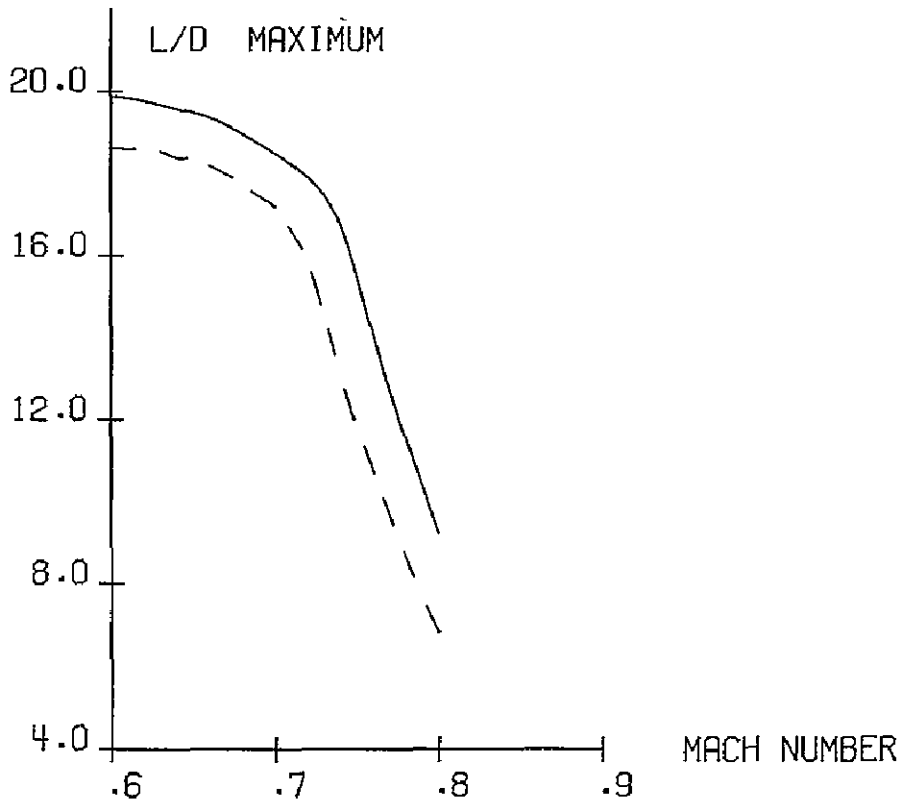
DRAG POLARS FOR AIRFOIL 75-07-15 AT $M=0.76$

FIGURE 29. ACCURACY OF NCF METHOD FOR CALCULATING AIRFOIL PERFORMANCE

D. Figure 30

Our supercritical airfoil 70-10-13 (see Volume II) was used by R. T. Jones in the design of a model of his oblique wing SST tested at the NASA Ames Research Center. An identical test was performed with a more conventional wing section. In both experiments transition was fixed by trips. For a straight configuration, i.e. with zero yaw, the data that were obtained provide an interesting experimental analysis of our design method. Figure 30 compares the maximum lift-drag ratios L/D for the two wings in their dependence on the Mach number M . In the transonic range the supercritical wing performed better, as was to be expected. More striking is the fact that for free stream Mach number $M = .6$, where the whole flow was subsonic, the maximum L/D for our airfoil was 7% higher than that for the comparison section. This is the best evidence currently available that our trailing edge model with a Stratford pressure distribution delivers superior performance (see Section 1 of Chapter III).

— SUPERCRITICAL WING 70-10-13
-- JONES COMPARISON WING



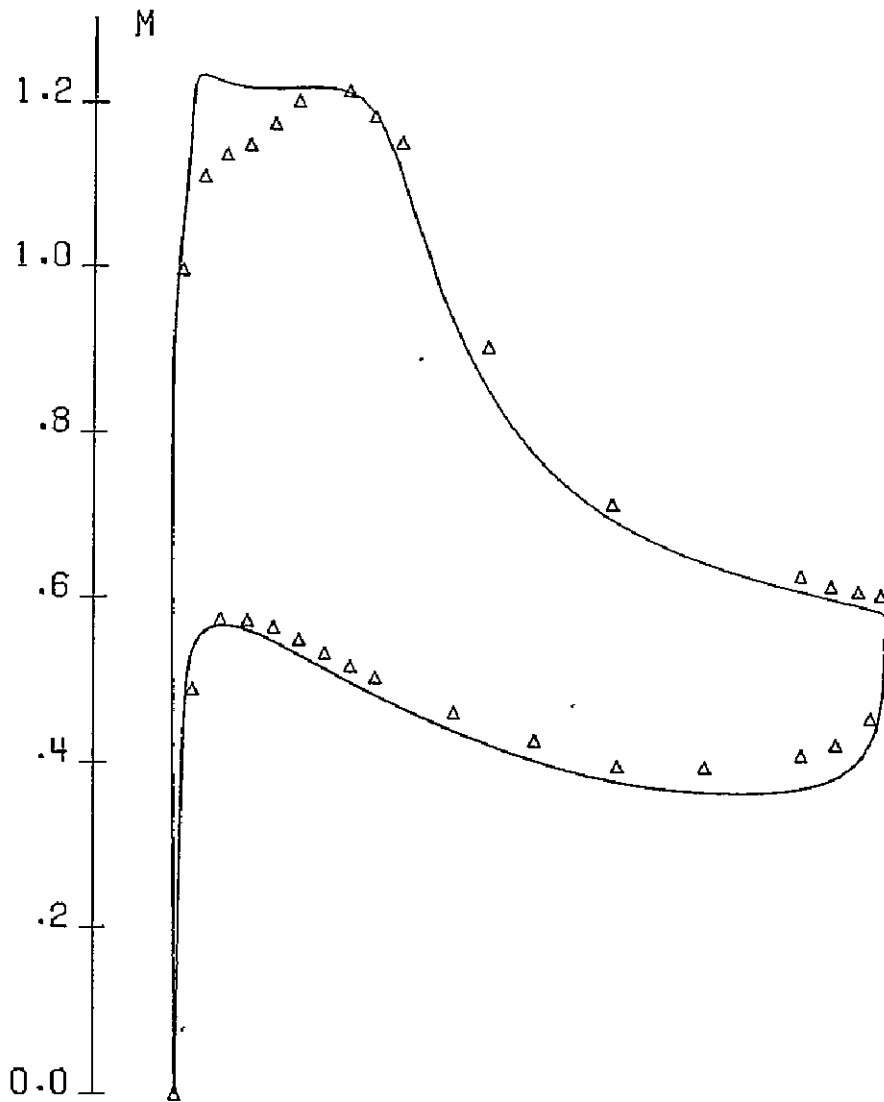
COMPARISON OF MAXIMUM LIFT/DRAG RATIO FOR A
CONVENTIONAL WING AND A SUPERCRITICAL WING IN
THREE DIMENSIONAL TRANSONIC FLOW WITHOUT YAW

FIGURE 30. JONES DATA DEMONSTRATING PERFORMANCE OF SUPERCRITICAL WING

E. Figure 31

In 1974 we designed a shockless compressor airfoil for the Pratt and Whitney Aircraft Division of United Technologies Corporation which has been tested by Harry Stephens at a cascade wind tunnel of the DFVLR in Germany. The blade was so heavily aft loaded that its thickness flaired out to 3% at the trailing edge after thinning down to only 2% at about 90% of chord. A similar airfoil based on the same pressure distribution is shown in Figure 8.

Pratt and Whitney Aircraft has given us permission to release the test data presented in Figure 31. This shows the experimental pressure distribution most closely fitting design. Following work of David Ives and Harry Stephens, we have made a stream tube correction that reduces the experimental Mach numbers along the profile by .04. The discrepancy between theory and experiment seen near the leading edge is due to an error in the model geometry. Accurate reproduction in the test of the aft loading defined by the design calculation confirms the theory of the Stratford pressure distribution described in Section 1 of Chapter III. Excellent performance of the airfoil in transonic conditions bodes well for supercritical wing technology as a tool to improve the efficiency of turbomachinery. Also encouraging was the performance of the supercritical blade over a wide range of subsonic flow conditions, where the loss coefficients ran as low as .016 .



PRATT AND WHITNEY COMPRESSOR AIRFOIL $R=1.1$ MILLION
 — THEORY $M_1=.780$ $M_2=.480$ DEL TH=25.0
 Δ EXPERIMENT $M_1=.775$ $M_2=.544$ DEL TH=25.5 LOSS=.0196

FIGURE 31. STEPHENS EXPERIMENTAL TEST OF SHOCKLESS CASCADE

VIII. FORTRAN LISTINGS OF THE CODES

1. THE NEW DESIGN CODE K

```

C      PROGRAM DESIGN (OUTPUT,TAPE4=OUTPUT,TAPE7=102B,TAPE1,TAPE3)
C      CALLS SUBROUTINES FOR DESIGN CODE
      DIMENSION CARD(81),DXDY(2)
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /G/ N1,N3,N4,N7,M1
      REAL MACHA,MACHB,MTAIL
      COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
      COMPLEX U,V,F1,F2,F3,S1,S2,S3,LAMDAP,LAMDAM,XI,ETA,TAU,PHI,PSI
      COMMON PHI(585,1),SKA(390),PSI(585,1),SKB(390),XI(585),ETA(585),
1 U(585),V(585),F1(585),S1(585),F2(585),S2(585),F3(585),S3(585),
2 LAMDAP(585),LAMDAM(585),TAU(585)
      DATA N1/1/ , N3/3/ , N4/4/ , N7/7/ , KBM/16/ , LBM/585/ , MX/1/
C      CHANGE LBM WHEN CHANGING THE DIMENSION SIZE
C      DIMENSION FOR SKA AND SKB IS 2*LBM/3
C      THE TOLERANCE DEPENDS ON THE PRECISION
      TOL = 1.E-12
      PI = ACOS(-1.)
      M1 = N1
      REWIND N1
      ISW = 1
C      SET ISW=0 IF N1 IS EMPTY
      READ (N1) NRNX
      IF (EOF(N1).NE.0.) ISW = 0
      REWIND N1
      REWIND N3
      REWIND N7
C      READ IN THE TAPE7 DATA
10 READ (N7,140) (CARD(J) , J = 1,80)
      IF (INC0DE(CARD).NE.0) GO TO 10
C      READ IN THE PRESSURE DISTRIBUTION
      IF (ISW.EQ.0) CALL READQS (GAM,PHMN,NRN)
      CALL TITLE
      IF ((ISW.NE.0).AND.(IABS(NRNX).NE.IABS(NRN))) GO TO 120
C      MAKE INITIAL GUESS OF PHI BASED ON INCOMPRESSIBLE FLOW
C      CHECK FOR EMPTY FILE. IF EMPTY DO INCOMPRESSIBLE GUESS FOR PHI
      IF (ISW.EQ.0) CALL PHIINC (GAM,PHMN)
C      MRP IS NEGATIVE WHEN CHOOSING THE THIRD ORDER ACCURATE METHOD
      MRPOLD = MRP
      NI = MAX0(1,NJ)
      IF (NJ.LE.0) IPLT = -IABS(IPLT)
C      IF TAPE1 IS NOT EMPTY AND NO CYCLES ARE REQUESTED THEN PLOT ONLY
      MODE = -10
      KBM = MIN0(KBM,(NF+1)/3)
      IF (ISW*IPLT.LT.0) GO TO 110

```



```

WRITE (N4,160)
C SECOND IS A CDC 6600 CENTRAL PROCESSOR TIMING ROUTINE
CALL SECOND(TIME)
MODE = 0
DO 60 N = 1,N1
CALL CYCLE(NZ)
MRP = 1
IF (N.EQ.N1) MRP = IABS(MRPOLD)
N1 = M1
20 CALL INIT (XI,ETA,U,V)
IF (IPLT.LT.0) GO TO 110
DO 40 MODE = 1,NP
IPASS = 1
DO 30 J = 2,NF,KBM
KK = 0
CALL GTPATH (XI,NC+1,NN,1)
II = JJ
NJ = J-2
NX = MINO(KBM,NF+1-J)
IPASS = IPASS+1
30 CALL MAIN (-J,IPASS)
40 CONTINUE
IF ((MRPOLD.NE.-MRP).OR.(MRPOLD.GT.0).OR.(N.NE.N1)) GO TO 50
C HALVE THE MESH AND TRY AGAIN
MRP = MRP+MRP
N1 = N3
REWIND N1
GO TO 20
50 CALL AUTO2(RES,DXDY)
EMA=ABS(MACHA)
EMB=ABS(MACHB)
60 WRITE(N4,170) NZ,EMB,EMA,RES,DXDY(1),DXDY(2)
CALL SECOND (T1)
TIME = T1-TIME
MODE = 99
CALL GTPATH (XI,1,1,1)
MSG = 10HAUTOMATION
WRITE (N4,130) MSG,JJ
WRITE (N4,150) MSG,TIME
MRP = IABS(MRPOLD)
N1 = M1
NX = 1
NJ = 0
70 DO 80 J = 1,9
CALL INIT (XI,ETA,U,V)
KK = 0
MODE = -J
CALL MAIN(J,1)
IF (MODE.EQ.-10) GO TO 90
80 CONTINUE
90 IF ((MRPOLD.GT.0).OR. (MRPOLD.NE.-MRP)) GO TO 100
MRP = MRP+MRP
N1 = N3
REWIND N1
GO TO 70
100 MSG = 10HSUPERSONIC
WRITE (N4,130) MSG,JJ

```

```

CALL SECOND (TIME)
TIME = TIME-T1
WRITE (N4,150) MSG,TIME
ENDFILE N1
IF (N1.NE.M1) ENDFILE M1
ISW = 1
110 CALL GOPLT (NRN,11HDAVID KORN ,10)
CALL CHNMDE
IF (ISW.NE.0) MX = NK+2
IF (MRPOLD.LT.0) N1 = N3
MRP = IABS(MRPOLD)
CALL BODYPT(MX,NPTS)
IPLT = IABS(IPLT)
CALL BLADE (MX,NPTS,CARD)
IF (IPLT.EQ.0) CALL EXIT
CALL HOGRF(MX,CARD)
CALL ENDPLT
CALL EXIT
120 WRITE (N4,180) NRNX
CALL EXIT
130 FORMAT(///13X8HLONGEST ,A10,10H PATH HAS ,I3,7H POINTS)
140 FORMAT (80A1)
150 FQRMAT(/13X,A10,11H CP TIME IS ,F7.1,8H SECONDS )
160 FORMAT(///13X5HCYCLE,4X6HMINLET,5X5HMEXIT,5X8HRESIDUAL,8X2HDX,8X2
1HDY/)
170 FORMAT(1H0,11XI5,1X,2F10.3,E13.3,1X,2F10.5)
180 FORMAT (/10X,27H****ATTEMPT TO CONTINUE RUN,I5,12H WITH WRONG
1 15HRUN NUMBER**** )
END

```

```

FUNCTION INCODE (DATA)
C READS IN THE FREE-FORM INPUT AND STORES THE VALUES
C VARLST CONTAINS THE LIST OF INPUT PARAMETERS
C LOCV CONTAINS THE ADDRESSES OF THE VARIABLES
C NEGATIVE ADDRESSES ARE USED FOR INTEGER VARIABLES
DIMENSION DATA(81),CHAR(39),VALUE(20),VVAL(2),VARLST(30),VARNAM(7)
1 ,LOCV(30),ARRAY(1),IARRAY(1)
COMPLEX ETJ
REAL MACH
COMMON /A/ GAMMA,MACH,ROOT(4)
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMMON /E/ ETJ(64),BUMP(5,10),NBMP
COMMON /G/ N1,N3,N4,N7,M1
COMPLEX XITAIL,BP,XIA,XIB,XIC
COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
REAL MACHA,MACHB,MTAIL
COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
COMMON /L/ PLTSZ,NPTS
EQUIVALENCE (ARRAY(1),IARRAY(1)),(RXIA,XIA)
DATA CHAR /1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,
1 1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HE,1HO,1HI,1H2,
2 1H3,1H4,1H5,1H6,1H7,1H8,1H9,1H+,1H-,1H./
DATA VARLST(1)/3HEND/ , VARLST(2)/2HNPP/ , VARLST(3)/3HNRN/
DATA VARLST(4)/3HNFC/ , VARLST(5)/4HKONE/ , VARLST(6) /3HMRP/

```

```

DATA VARLST(7)/4HKTWO/ ,VARLST(8)/2HNF/ , VARLST(9) /2HNI /
DATA VARLST(10)/4HIPLT/ , VARLST(11)/2HRN/ , VARLST(12)/3HXIA/
DATA VARLST(13)/5HPLTSZ/ , VARLST(14)/3HXIB/ , VARLST(15)/3HXIC/
DATA VARLST(16)/5HGAMMA/ , VARLST(17)/5HTRANU/ ,VARLST(18)/4HGRID/
DATA VARLST(19)/6HMINLET/ ,VARLST(20)/5HMEXIT/ ,VARLST(21)/4HMACH/
DATA VARLST(22)/7HANGEXIT/ ,VARLST(23)/5HANGIN/ ,VARLST(24)/4HKTHR/
DATA VARLST(25)/5HTRANL/ ,VARLST(26)/3HRUN/ , VARLST(27)/4HBUMP/
DATA VARLST(28)/4HNPTS/
DATA NCRD/0/ , NLST /28/ , BL /1H / ,STAR /1H*/ , K/1/ , ICMP/2/
LOCV(NLST+1) = LOCF(LOCV(NLST+1))
NX = 1
KX = 1
MODE = 1
IF (NCRD.NE.0) GO TO 10
LOCV(2) =-LOCF(NP)
NP = 8
LOCV(3) =-LOCF(NRN)
NRN = 1
LOCV(4) =-LOCF(NFC)
NFC=64
LOCV(5) = LOCF(CONE)
CONE = .5
LOCV ( 6) = -LOCF(MRP)
MRP=-1
LOCV(7) = LOCF(CTWO)
CTWO = 1.
LOCV(8) = -LOCF(NF)
NF = 0
LOCV(9) = -LOCF(NJ)
NJ = 1
LOCV(10) = -LOCF(IPLT)
IPLT = 37
LOCV(11) = LOCF(RN)
RN = 0.
LOCV(12) = LOCF(RXIA)
XIA = (.08,-.12)
LOCV(13) = LOCF(PLTSZ)
PLTSZ = 0.
LOCV(14) = LOCV(12) + ICMP
XIB = XIA
LOCV(15) = LOCV(14) + ICMP
XIC=(2.,0.)
LOCV(16) = LOCF(GAMMA)
GAMMA = 1.4
LOCV(17) = LOCF(TRANU)
TRANU = .05
LOCV(18) = LOCF(GRID)
GRID = .08
LOCV(19) = LOCF(MACHB)
MACHB = 0.
LOCV(20) = LOCF(MACHA)
MACHA = 0.
LOCV(21) = LOCF(MACH)
MACH = .75
LOCV(22) = LOCF(ANGLA)
ANGLA = -18.
LOCV(23) = LOCF(ANGLB)

```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```

    ANGLB = 7.
    LOCV(24) = LOCF(CTHR)
    CTHR = 1.
    LOCV(25) = LOCF(TRANL)
    TRANL = .10
    LOCV(26) = -LOCF(NRN)
    LOCV(27) = LOCF(BUMP(1))
    LOCV(28) = -LOCF(NPTS)
    NPTS = 201
    NBMP = 0
10  NCRD = NCRD+1
    DO 20 L = 1,7
20  VARNAM(L) = 1H
    SAVE = DATA(81)
    DATA (81) = 1H$
    DO 300 L = 1,81
C   IGNORE BLANKS
    IF (DATA(L).EQ.1H ) GO TO 300
C   RESET ON ENCOUNTERING A DOLLAR SIGN
    IF ((DATA(L).EQ.1H$).OR.(DATA(L).EQ.1H;)) GO TO 60
    IF (NX.LT.0) GO TO 300
    IF (DATA(L).EQ.1H= ) GO TO 150
C   FIND OUT WHICH CHARACTER WE HAVE
    DO 30 J = 1,39
    IF (DATA(L).EQ. CHAR(J)) GO TO 40
30  CONTINUE
    J = 40
40  IF (NX.EQ.0) GO TO 190
    IF (J.GT.26) GO TO 270
    VARNAM(NX) = DATA(L)
    NX = NX+1
    IF (NX.GT.8) GO TO 280
    GO TO 300
C   CHECK FOR END OF DATA
60  IF (NX.LE.1) GO TO 70
    PRINT 410,BL,VARNAME
    ENCODE (10,410,VNAME) VARNAM
    IF (VARLST(1).EQ.VNAME) GO TO 310
C   SET VARNAM BACK TO BLANK
70  DO 80 J = 1,7
80  VARNAM(J) = 1H
    J = NX
    NX = 1
    IF ( J.NE.0) GO TO 300
90  IF (J.EQ.0) NX = 1
    LOCV(K) = ISIGN(IABS(LOCV(K))+1,LOCV(K))
    IF (KX.EQ.1) GO TO 290
    IF (KX.GT.20) GO TO 110
C   PUT NUMBER INTO VVAL RIGHT JUSTIFIED
    ENCODE (20,410,VVAL) (BL,J = KX,20),(VALUE(J-1),J = 2,KX)
    KX = 1
    IF (MODE.GT.2) GO TO 110
    KA = IABS(LOCV(K))-LOCF(ARRAY(1))
    IF (MODE.EQ.2) GO TO 100
C   INTEGER CONVERSION
    DECODE (20,420,VVAL) IARRAY(KA)
    IF (LOCV(K).GT.0) ARRAY(KA) = IARRAY(KA)

```

```

      GO TO 300
C     CONVERT TO REAL
100  DECODE (20,430,VVAL) ARRAY(KA)
      IF (LOCV(K).LT.0) IARRAY(KA) = ARRAY(KA)+.5
      MODE = 1
      GO TO 300
110  MODE = 1
      GO TO 300
150  NX = 0
      ENCODE (10,410,VNAME) VARNAM
C     LOOK FOR THE VARIABLE ON THE LIST
      DO 160 K = 1,NLST
      IF (VARLST(K).EQ.VNAME) GO TO 300
160  CONTINUE
      PRINT 450, VNAME
      GO TO 280
190  IF((J.LE.25).OR.(J.EQ.40)) GO TO 200
C     CHANGE THE MODE UPON ENCOUNTERING A DECIMAL POINT
      IF (DATA(L).EQ.1H.) MODE = MODE+1
C     SAVE THE DIGITS TO FORM A NUMBER
      VALUE(KX) = DATA(L)
      KX = KX+1
      GO TO 300
C     CHECK FOR A DECIMAL POINT
C     CHECK FOR A COMMA
200  IF (DATA(L).EQ.1H.) GO TO 90
      MODE = 3
      KX = KX+1
      GO TO 300
270  PRINT 460
280  PRINT 440, NCRD,(DATA(J),J = 1,80)
      LP = L+35
      PRINT 410, (BL, J = 1,LP),STAR
C     SET ERROR FLAG AND LOOK FOR NEXT DOLLAR SIGN
      NX = -1
290  MODE = 1
300  CONTINUE
310  DATA(81) = SAVE
      INCODE = K-1
      IF (INCODE.NE.0) RETURN
      IF((AIMAG(XIC).EQ.0.).AND.(REAL(XIC).NE.2.))GO TO 330
      IF(XIA.NE.XIB)GO TO 320
      IF(REAL(XIC).EQ.2.) XIC=XIB-(.1,.1)
      IF(REAL(XIC).EQ.0.) XIC=XIB+ABS(AIMAG(XIB-XIC))*CMPLX(COS(.75*PI),
1-SIN(.75*PI))
      GO TO 330
320  IF(REAL(XIC).EQ.2.) XIC=XIB+.07*(XIB-XIA)/CABS(XIB-XIA)
      IF(REAL(XIC).EQ.0.) XIC=XIB+ABS(AIMAG(XIB-XIC))*(XIB-XIA)/CABS(XIB
1-XIA)
330  CONTINUE
C     RESTORE LINE COUNTER
      NCRD = 0
      NK = NFC+NFC
      NBPS = NFC+1
      IF (NF.LE.0) NF = NK
      N6MP = (LOCV(27)-LOCF(BUMP(1)))/5
      RETURN

```

```

410 FORMAT (80A1)
420 FORMAT (I20)
430 FORMAT (E20.0)
440 FORMAT (19H0****ERROR IN CARD ,I3,5H ****, 9X,80A1)
450 FORMAT (*OVARIABLE NAMED *,A8, * WAS NOT FOUND*)
460 FORMAT(*ONON-ALPHABETIC CHARACTERS NOT ALLOWED IN VARIABLE NAMES*)
END

```

```

C      SUBROUTINE READQS(GAM,PHMN,NRN)
      SUBROUTINE TO READ IN THE INPUT PRESSURE DISTRIBUTION
      COMPLEX ETJ
      COMMON /E/ ETJ(64),BUMP(5,10),NBMP
      COMMON /G/ N1,N3,N4,N7,M1
      COMMON /L/ PLTSZ,NPTS
      COMMON AX(130),BX(130),SS(130),QLG(130),SI(300),QI(300),ZERO(300),
1      1DQDS(300),DPHDS(300),FP(300),FPP(300),FPPP(300),QPP(300),QPPP(300)
2      ,ES(600),Q(600),PHI(600),PHII(130)
      COMMON SX(300),QX(300)
      DATA NINMAX /300/
      WRITE (N4,150) NRN
      READ (N3,110) XIN
      NIN = XIN
      IF (NIN.GT.NINMAX) GO TO 90
C      READ IN Q(S)
      DO 20 J = 1,NIN
      READ (N3,120) SI(J),QI(J)
      SX(J) = SI(J)
C      MODIFY THE INPUT PRESSURE DISTRIBUTION
      QMOD = 1.
      IF(NBMP.EQ.0) GO TO 20
      DO 10 L = 1,NBMP
10      QMOD = (1.+BUMPFN(SI(J),BUMP(1,L),BUMP(3,L),BUMP(4,L),BUMP(2,L),
1      1 BUMP(5,L)))*QMOD
20      QX(J)=QI(J)*QMOD
      ARCL = SX(NIN)-SX(1)
      FAC = 2./ARCL
      CONST = -1.-FAC*SX(1)
      WRITE(N4,160)
      WRITE (N4,130)
      DO 30 J = 1,NIN
      SX(J) = FAC*SX(J)+CONST
      WRITE (N4,140) J,SI(J),QI(J),SX(J),QX(J)
      QI(J)=QX(J)
30      CONTINUE
      SX(NIN) = 1.
      WRITE (N1) NRN,NIN,(SX(J),QX(J), J = 1,NIN)
      DS =(SX(NIN)-SX(1) )/FLOAT(NPTS-1)
C      FIND Q(S) AT EVENLY SPACED POINTS
      ES(1) = SX(1)
      DO 40 J = 2,NPTS
40      ES(J) = ES(J-1)+DS

```

```

ES(NPTS) = SX(NIN)
CALL SPLIF(NIN, SX, QX, FP, FPP, FPPP, 3, 0., 3, 0.)
CALL INTPL(NPTS, ES, Q, SX, QX, FP, FPP, FPPP)
: INTEGRATE Q(S) TO GET PHI(S)
CALL SPLIF(NPTS, ES, Q, DQDS, FPP, PHI, -3, 0., 3, 0.)
GAM = PHI(NPTS)
: SPLINE FIT PHI AS A FUNCTION OF S
CALL SPLIF(NPTS, ES, PHI, DPHDS, FPP, FPPP, 3, 0., 3, 0.)
: SPLINE FIT Q AS A FUNCTION OF S
CALL SPLIF(NPTS, ES, Q, DQDS, QPP, QPPP, 3, 0., 3, 0.)
: FIND S WHERE Q VANISHES
CALL INTPL(1, SO, 0., NPTS, ES, Q, DQDS, QPP, QPPP)
: FIND MINIMUM VALUE OF PHI
CALL INTPL(1, SO, PHMN, ES, PHI, DPHDS, FPP, FPPP)
DO 50 I = 1, NPTS
VAL = AMAX1(0., PHI(I) - PHMN)
50 PHI(I) = SIGN(SQRT(VAL), ES(I) - SO)
WRITE (N1) NPTS, SO, PHMN, GAM, (ES(J), Q(J), PHI(J), J = 1, NPTS)
RETURN
90 WRITE (N4, 100) NINMAX
CALL EXIT
100 FORMAT (15H0****MORE THAN ,14,30H INPUT CARDS NOT PERMITTED**** /
1 32H0***PROGRAM STOPPED IN READQS*** )
110 FORMAT(F5.0)
120 FORMAT (4E20.8)
130 FORMAT(1H0/11X,4HCARD,5X,7HS-INPUT,8X,7HQ-INPUT,10X,6HS-USED
1 ,9X,6HQ-USED /)
140 FORMAT(3X,19,2F15.6,3X,2F15.6)
150 FORMAT(1H1/8X,22HBEGIN EXECUTION OF RUN ,14)
160 FORMAT(1H0/38X,13HTAPE3 = INPUT/)
END

```

```

C SUBROUTINE TITLE
PRINTS THE TITLE PAGE OF THE OUTPUT
REAL MACH
COMMON /A/ GAMMA, MACH
COMMON /C/ PI, GRID, TOL
COMMON /D/ II, IPLT, JJ, KBM, KK, LBM, MODE, MRP, NB, NC, NF, NJ, NK, NN, NP, NX
COMPLEX ETJ
COMMON /E/ ETJ(64), BUMP(5, 10), NBMP
COMMON /G/ N1, N3, N4, N7, M1
COMPLEX XITAIL, BP, XIA, XIB, XIC
COMMON /H/ BP(129), XIA, XIB, XIC, XITAIL, CONE, CTWO, CTHR, RATC, QR, NBPS
REAL MACHA, MACHB, MTAIL
COMMON /K/ MACHA, MACHB, MTAIL, ANGLA, ANGLB, ANGLT, RN, TRANU, TRANL, NRN
COMMON /L/ PLTSZ, NPTS
C - DATE IS A CDC ROUTINE WHICH GIVES THE MONTH, DAY, AND YEAR
CALL DATE(IDATE)
NRUN = IABS(NRN)
TYPE = 7HCASCADE
IF (CABS(XIA - XIB).LT.TOL) TYPE = 7HAIRFOIL
WRITE (N4, 80) TYPE, NRUN, IDATE
NM1 = 8H RUN =

```

```

NM2 = 8H   MRP =
WRITE (N4,60) NM1,NRN,NM2,MRP
NM1 = 8H   MACH =
NM2 = 8H   RN =
E6 = 2HE6
RNX = 1.E-6*RN
IF (RNX.EQ.0.) E6 = 1H
WRITE (N4,70) NM1,MACH,NM2,RNX,E6
IF (RN.EQ.0.) GO TO 10
NM1 = 8H TRANU =
NM2 = 8H TRANL =
WRITE (N4,90) NM1,TRANU,NM2,TRANL
10 NM1 = 8H KONE =
NM2 = 8H KTWO =
WRITE (N4,90) NM1,CONE,NM2,CTWO
NM1 = 8H KTHR =
NM2 = 8H NPTS =
WRITE (N4,100) NM1,CTHR,NM2,NPTS
NM1 = 8H   NF =
NM2 = 8H   NFC =
NFC = NK/2
WRITE (N4,60) NM1,NF,NM2,NFC
NM1 = 8H GAMMA =
NM2 = 8H GRID =
WRITE (N4,90) NM1,GAMMA,NM2,GRID
NM1 = 8H   NI =
NM2 = 8H   NP =
WRITE (N4,60) NM1,NJ,NM2,NP
NM1 = 8H PLTSZ =
NM2 = 8H IPLT =
WRITE (N4,100) NM1,PLTSZ,NM2,IPLT
NM1 = 8H   XIA =
NM2 = 8H   XIB =
WRITE (N4,130) NM1,XIA,NM2,XIB
NM1 = 8H   XIC =
WRITE (N4,150) NM1,XIC
C WRITE OUT BUMPS
IF (NBMP.LE.0) RETURN
DO 50 J = 1,NBMP
50 WRITE (N4,140) J,(BUMP(L,J), L = 1,5)
RETURN
60 FORMAT(/17X,A8,I4,13X1H;12XA8,I4)
70 FORMAT(/17X,A8,F6.3,11X1H;12XA8,F5.1,A2)
80 FORMAT (1H1/15X,10HTRANSONIC ,A8,10HDESIGN RUN I5,17XA10///37X,
1 12HTAPE7=INPUT /)
90 FORMAT(/17X,A8,F6.2,11X1H;12XA8,F6.2)
100 FORMAT(/17X,A8,F6.2,11X1H;12XA8,I4 )
130 FORMAT (/9XA8,F6.3,2H ,F6.3,11X1H;4XA8,F6.3,2H ,F6.3)
140 FORMAT (1H0,9X,5HBUMP(,I1,3H) =,5F10.5)
150 FORMAT(/27X,A8,F6.3,2H ,F6.3/)
END

```



```

SUBROUTINE PHIINC (GAM,PHMN)
C   CALCULATES VELOCITY POTENTIAL PHI FOR INCOMPRESSIBLE FLOW
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMMON /G/ N1,N3,N4,N7,M1
COMPLEX XITAIL,BP,XIA,XIB,XIC
COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
COMPLEX XIP,XIPP,TEMP,CLOG,ROT,XIX,XI,A,XIO
COMMON PHIWO(3),PHIWT(3),PHII(128)
COMPLEX ZED
DATA QR/1./ , ITER /0/ , ZERO/0./
ROT = (1.+CONJG(XIB))/(1.+XIB)
XIP = ROT*(XIA-XIB)/(1.-CONJG(XIB)*XIA)
XIPP = XIP+1.
AXIPSQ = REAL(XIP)**2+AIMAG(XIP)**2
C   EVALUATE PHI AND DPHI/DW AT STAGNATION FOR EACH SINGULAR TERM
TEMP = -1.
IF (AXIPSQ.NE.0.) TEMP = -CLOG(XIPP)/XIP
PHIWO(1) = 2.*REAL(TEMP)
PHIWO(2) = -2.*AIMAG(TEMP)
PHIWO(3) = 2.*ATAN2(AIMAG(XIPP),REAL(XIPP)) - PI
CF = GAM/(PI+PI)
WELL = GAM-PHMN
C   THE FIRST EQUATION REQUIRES DPHI/DW=0 AT STAGNATION
TEMP = -2./XIPP
A11 = AIMAG(TEMP)
A12 = REAL(TEMP)
B1 = -CF*(1.-AXIPSQ)/(REAL(XIPP)**2+AIMAG(XIPP)**2)
C   WT IS CIRCLE ANGLE CORRESPONDING TO THE TRAILING EDGE
XITAIL=ROT
WT=ATAN2(AIMAG(XITAIL),REAL(XITAIL))
C   ITERATE AT MOST 20 TIMES TO FIND WT
DO 20 J = 1,20
WTOLD = WT
C   EVALUATE PHI AT WT FOR EACH SINGULAR SOLUTION
TEMP = CONJG(XITAIL)
IF (AXIPSQ.NE.0.)TEMP = -CLOG(1.-TEMP*XIP)/XIP
PHIWT(1) = 2.*REAL(TEMP)
PHIWT(2) = -2.*AIMAG(TEMP)
PHIWT(3)=2.*ATAN2(AIMAG(1.-XIP/XITAIL),REAL(1.-XIP/XITAIL))+WT
A21 = PHIWT(1) - PHIWO(1)
A22 = PHIWT(2)-PHIWO(2)
B2 = -CF*(PHIWT(3)-PHIWO(3))+WELL
AF = (A22*B1-A12*B2)/(A11*A22-A21*A12)
BF = (B1-A11*AF)/A12
C   NOW COMPUTE THE LOCATION OF THE TRAILING EDGE
XITAIL = CMPLX(AF,BF)/CMPLX(AF,-BF)
WT = ATAN2(AIMAG(XITAIL),REAL(XITAIL))
IF (ABS(WT-WTOLD).LT.TOL) GO TO 30
20 CONTINUE
30 A = CMPLX(AF,BF)
C   FIND W AT THE TRAILING EDGE
XI = CONJG(ROT)*XITAIL
XI = (XI+XIB)/(1.+XI*CONJG(XIB))
W = ATAN2(AIMAG(XI),REAL(XI))
WTAIL = W
C   FIND PHI AT EVENLY SPACED POINTS STARTING FROM STAGNATION

```

```

CONST = AF*PHIWO(1)+BF*PHIWO(2)+CF*PHIWO(3)
DW = (PI+PI)/FLOAT(NK)
ARG=-PI
W = -PI
XIX = -1.
DO 40 J = 1,NK
XI = CMPLX(COS(W),SIN(W))
XIO = ROT*(XI-XIB)/(1.-CONJG(XIB)*XI)
TEMP = CONJG(XIO)
IF (AXIPSQ.NE.0.) TEMP = -CLOG(1.-TEMP*XIP)/XIP
AT1=ATAN2(AIMAG(XIX/XIO),REAL(XIX/XIO))
XIX = XIO
ARG=ARG-AT1
PHII(J)=2.*REAL(A*TEMP)+CF*ARG-CONST+2.*CF*ATAN2(AIMAG(1.-XIP/XIO)
1,REAL(1.-XIP/XIO))
W = W + DW
40 CONTINUE
NT = NK+1
PHII(NT) = GAM
WRITE (N1) QR,QR,ZERO,ITER,NT,(PHII(J),ZERO, J = 1,NT)
WRITE (N1) WTAIL
RETURN
END

```

```

C SUBROUTINE INIT (XI,ETA,U,V)
INITIALIZE THE PATH FROM INFINITY TO XIC
COMPLEX I,S,H,SOFXI,XIOFM,ROOT,ROOT1,XI(1),ETA(1),U(1),V(1)
REAL MACH
COMMON /A/ GAMMA,MACH,ROOT,ROOT1
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMMON /F/ UOLD,VOLD,SINGF,SINGS,SINGX,SINGY,PHIOLD(20),PSIOLD(20)
1 , XOLD(20),YOLD(20),XAIMG,YAIMG,THLAST,CL,DF
COMPLEX XITAIL,BP,XIA,XIB,XIC
COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWB,CTHR,RATC,QR,NBPS
REAL MACHA,MACHB,MTAIL
COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
LOGICAL ISW
DATA I/(0.,1.)/ , ISW/.TRUE./ , ROOT/(0.,1.)/ , ROOT1/(1.,0.)/
ROOT = (0.,1.)
ROOT1 = (1.,0.)
IF (ISW) GO TO 5
IF (MACHA.GT.0.) XIA = XIOFM(MACHA,ANGLA,XIA)
IF (MACHB.GT.0.) XIB = XIOFM(MACHB,ANGLB,XIB)
5 ISW = .FALSE.
C LAY DOWN PATH FROM XIA TO XIB TO XIC IN ETA-PLANE
XI(1) = XIA
C COMPUTE THE NUMBER OF POINTS FROM XIA TO XIB
MM = CABS(XIB-XIA)/GRID + 1.-TOL
MM = MM*MRP
NB = MM+1
IF (NB.EQ.1) GO TO 15
H = (XIB-XIA)/FLOAT(MM)

```

```

C      LAY DOWN GRID TO XIB
      DO 10 L = 2,NB
10     XI(L) = XI(L-1)+H
C      COMPUTE THE NUMBER OF POINTS FROM XIB TO XIC
15     MM = CABS(XIB-XIC)/GRID + 1.-TOL
      MM = MM*MRP
      NC = MM+NB
      H = (XIC-XIB)/FLOAT(MM)
C      LAY DOWN GRID FROM XIB TO XIC
      NBP = NB+1
      DO 20 L = NBP,NC
20     XI(L) = XI(L-1)+H
      IF (MODE.EQ.-10) RETURN
C      LAY DOWN REFLECTED GRID IN THE XI-PLANE
      DO 30 L = 1,NC
30     ETA(L) = CONJG(XI(L))
      RAD = PI/180.
      GAM = (GAMMA-1.)/2.
      COSQ = 1./(MACH*MACH)+GAM
      IF (MACHB.GT.0.) GO TO 40
C      FIND MACH NUMBER AND ANGLE AT INLET
      S = SDFXI(XIB)
      U(NB) = 1.
      V(NB) = 0.
      CALL GETUV (S,CONJG(S),U(NB),V(NB))
      UB = REAL(U(NB))
      VB = REAL(V(NB))
      QSB = UB*UB+VB*VB
      MACHB = -SQRT(QSB/(COSQ-GAM*QSB))
      ANGLB = ATAN2(VB,UB)/RAD
      GO TO 45
40     QSB = COSQ*MACHB*MACHB/(1.+GAM*MACHB*MACHB)
      UB = SQRT(QSB)*COS(RAD*ANGLB)
      VB = SQRT(QSB)*SIN(RAD*ANGLB)
      U(NB) = UB
      V(NB) = VB
45     IF (MACHA.GT.0.) GO TO 50
C      FIND MACH NUMBER AND ANGLE AT EXIT
      U(1) = 1.
      V(1) = 0.
      S = SDFXI(XIA)
      CALL GETUV (S,CONJG(S),U(1),V(1))
      UA = REAL(U(1))
      VA = REAL(V(1))
      QSA = UA*UA+VA*VA
      MACHA = -SQRT(QSA/(COSQ-GAM*QSA))
      ANGLA = ATAN2(VA,UA)/RAD
      RETURN
50     QSA = COSQ*MACHA*MACHA/(1.+GAM*MACHA*MACHA)
      UA = SQRT(QSA)*COS(RAD*ANGLA)
      VA = SQRT(QSA)*SIN(RAD*ANGLA)
      U(1) = UA
      V(1) = VA
      RETURN
      END

```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```

SUBROUTINE CYCLE (ITER)
C ITERATES TO FIND MAP FUNCTION
COMPLEX H,HP,SIG
REAL MACH
COMMON /A/ GAMMA,MACH
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMMON /G/ N1,N3,N4,N7,M1
COMPLEX XITAIL,BP,XIA,XIB,XIC
COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
REAL MACHA,MACHB,MTAIL
COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
COMMON AX(130),BX(130),SS(130),QLG(130),SI(300),QI(300),ZERO(300),
1DQDS(300),DPHDS(300),FP(300),FPP(300),FPPP(300),QPP(300),QPPP(300)
2,ES(600),Q(600),PHI(600),PHII(130)
COMMON QOLD(130)
DIMENSION DQDPH(300)
EQUIVALENCE (DQDPH(1),DPHDS(1))
LOGICAL ISW
DATA GX /1./, ISW/.FALSE./
IF (ISW) GO TO 10
TO = PI
NK2 = NK/2
DT = (PI+PI)/FLOAT(NK)
GAMP = (GAMMA+1.)/2.
GAMM = (GAMMA-1.)/2.
EMUSQ = GAMM/GAMP
QSTRSQ = EMUSQ+1./{(GAMP*MACH*MACH)}
10 ISW = .TRUE.
REWIND N1
READ (N1)
READ (N1) NPTS,SO,PHMN,GAM,(ES(J),Q(J),PHI(J), J = 1,NPTS)
READ (N1) QR,QS,RATC,ITER,NT,(PHII(J),QOLD(J), J = 1,NT)
IF (NT.NE.NK+1) GO TO 70
READ (N1) WTAIL
READ (N1) MACH
C SPLINE FIT Q AS A FUNCTION OF S
CALL SPLIF (NPTS,ES,Q,DQDS,QPP,QPPP,3,0.,3,0.)
DO 20 J = 1,NPTS
20 ZERO(J) = 0.
C SPLINE FIT Q AS A FUNCTION OF PHI
CALL SPLIF(NPTS,PHI,Q,DQDPH,FPP,FPPP,3,0.,3,0.)
C PHII IS THE GUESS FOR PHI AT EVENLY SPACED POINTS IN THE CIRCLE
PHII(1) = 0.
PHII(NT) = GAM
GAM2 = .5*GAM
DO 30 J = 1,NT
SS(J) = FLOAT(J-1)*DT-PI
FAC = SIGN(1.,WTAIL-SS(J))
VAL = AMAX1(0.,PHII(J)-(1.-FAC)*GAM2)
AX(J) = FAC*SQRT(VAL)
C COMPUTE QI AT THE CORRESPONDING POINTS
CALL INTPL(1,AX(J),QI(J),PHI,Q,DQDPH,FPP,FPPP)
30 CONTINUE
IF (ITER.GT.0) GO TO 40
C FIND DQ/DW AT THE STAGNATION POINT
DQDW=(QI(2)-QI(NT-1))/(2.*DT)

```

```

      CALL GETHSQ((0.,0.),H,HP)
      RATC= 1.-1./((ABS(DQDW)*SQRT(REAL(HP))))
40  ITER = ITER+1
      RAT = 1.+RATC*RATC
      SUM = 0.
C    GET COMPRESSIBLE ANALOG FOR LOG(Q)
      DO 50 J = 2,NK
      RFAC = GX-1.
      IF ((J.EQ.2).OR.(J.EQ.NK)) RFAC = .3*GX-1.
      IF (ITER.EQ.1) RFAC = 0.
      AX(J) = ABS(QI(J))-QOLD(J)
      QI(J) = ABS(QI(J))+RFAC*AX(J)
      K = J+NK2
      IF (K.GT.NK) K = K-NK
      CALL GETHSQ(CMPLX(QR*QI(J)*QI(J),0.),H,HP)
      T = SS(J)
      FAC = 4.*SIN(.5*(T-TD))**2/(RAT-2.*RATC*COS(T-TD))
      HIMG = ABS(AIMAG(H))*CTWD
      HAB = CABS(H)+HIMG*CONE
      QLG(K) = .5*ALOG(HAB/FAC)
      SUM = SUM+ QLG(K)
50  CONTINUE
      QLG(NK2+1) = QLG(NK2)+QLG(NK2+2)-.5*(QLG(NK2-1)+QLG(NK2+3))
      QLG(NT) = QLG(1)
      AVE = (SUM+QLG(NK2+1))/FLOAT(NK)
      SIG = CMPLX(EXP(AVE),0.)
      H=(1.0,0.0)
      HP=(0.,0.)
      CALL GETUV(SIG,CONJG(SIG),H,HP)
      QS = REAL(H)*REAL(H)+REAL(HP)*REAL(HP)
      QR = QR/QS
      MACH = SQRT(1./((GAMP*QR*QSTRSQ-GAMM))
C    COMPUTE THE COEFFICIENTS FOR THE MAPPING FUNCTION
      CALL FOUCF(NK,QLG ,BP,AX,BX)
      BP(1) = CMPLX(REAL(BP(1)),0.)
      BP(1) = 0.
      NFC = NK/2
C    FILTER THE FOURIER COEFFICIENTS
      FAC = (PI+PI)/FLOAT(NK)
      DO 60 J = 1,NFC
      SIG = SIN(FLOAT(J)*FAC)/(FLOAT(J)*FAC)
60  BP(J+1) = SIG*BP(J+1)
      H = SOFXI(XIC)
      H = SOFXI(XIA)
      REWIND N1
C    SKIP PAST FIRST TWO RECORDS ON TAPE1
      READ (N1)
      READ (N1)
      WRITE (N1) QR,QS,RATC,ITER,NT,(PHII(J),QI(J) , J = 1,NT)
      WRITE (N1) WTAIL
      RETURN
70  WRITE (N4,80) NK,NT
      CALL EXIT
80  FORMAT (1H09X,7H****NK=I3,8H AND NT=I3,21H ARE INCOMPATIBLE****
1 //10X32H****PROGRAM STOPPED IN CYCLE**** )
      END

```

```

      COMPLEX FUNCTION XIOFS(S,XIPAST)
C     CALCULATES CHARACTERISTIC COORDINATE XI AS A FUNCTION OF S
      COMPLEX XIPAST,XIOLD,XINEW,SOFXI,SOFXIP,S
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /G/ N1,N3,N4,N7,M1
      XIOLD = XIPAST
C     DO AT MOST 20 NEWTON ITERATIONS
      DO 40 K = 1,20
      XINEW = XIOLD-(SOFXI(XIOLD)-S)/SOFXIP(XIOLD)
      IF (CABS(XINEW-XIOLD).LE.TOL) GO TO 50
      XIOLD = XINEW
40    CONTINUE
      XINEW = SOFXI(XIOLD)-S
      WRITE (N4,60) S,XIOLD,XINEW
      XINEW=XIPAST
50    XIOFS = XINEW
      RETURN
60    FORMAT(/* NO CONVERGENCE AT S=*,2F6.3,5X3HXI=,2F6.3,6X,*S(XI)-S=*
1      , 2F6.3/* TROUBLE CALCULATING SONIC LINE. CHANGE MACH *)
      END

```

```

      COMPLEX FUNCTION SOFXI (XI)
C     CALCULATES HODOGRAPH VARIABLE S AS A FUNCTION OF XI
      COMMON /Q/ FP,TEMP
      COMPLEX XI,X,FN,FP,S,CEXP,TEMP
      COMPLEX XITAIL,BP,XIA,XIB,XIC
      COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
C     S = ((XI+1.)/ (RATC*XI+1.))*EXP(FN(XI))
C     WHERE FN(XI) = BP(J+1)*XI+J ; J = 0 TO NBPS-1
      DATA X /(1.E10,0.)/
      IF ((REAL(X).EQ.REAL(XI)).AND.(AIMAG(X).EQ.AIMAG(XI))) GO TO 20
      X = XI
      IF (REAL(X)**2+AIMAG(X)**2.GT.1.21) X = 1.10*X/CABS(X)
C     FIRST EVALUATE FN AND ITS DERIVATIVE
      FN = BP(NBPS)*X+BP(NBPS-1)
      FP = BP(NBPS)
      J = NBPS-2
10    FP = X*FP+FN
      FN = X*FN+BP(J)
      J = J-1
      IF (J.GE.1) GO TO 10
      FN = CEXP(FN)
      TEMP = 1./ (RATC*X+1.)
      S = (X+1.)*(TEMP*FN)
      TEMP = (1.-RATC)*TEMP*(TEMP*FN)
20    SOFXI = S
      RETURN
      END

```

```

COMPLEX FUNCTION SOFXIP(XI)
COMPLEX XITAIL,BP,XIA,XIB,XIC
COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
COMMON /Q/ FP,TEMP
COMPLEX XI,TEMP,FP,SOFXI
SOFXIP = SOFXI(XI)
SOFXIP = TEMP+FP*SOFXIP
RETURN
END

```

```

C      COMPLEX FUNCTION XIOFM(M,ANG,XIOLD)
      COMPUTE XI CLOSEST TO XIOLD WHICH HAS MACH NUMBER M AND ANGLE ANG
      COMMON /A/ GAMMA,MACH
      REAL M,MACH
      COMPLEX HSQ,HSQPR,S,XIOFS,XIOLD
      DATA EMOLD/0./ , PI/3.14159265358979/
      IF (MACH.EQ.EMOLD) GO TO 10
      EMOLD = MACH
      RAD = PI/180.
      GAM = (GAMMA-1.)/2.
      COSQ = 1./(MACH*MACH) + GAM
10    EMSQ = M*M
      ANGL = ANG*RAD
      QS = COSQ*EMSQ/(1.+GAM*EMSQ)
      CALL GETHSQ(CMPLX(QS,0.),HSQ,HSQPR)
      S = SQRT(REAL(HSQ))*CMPLX(COS(ANGL),-SIN(ANGL))
      XIOFM = XIOFS(S,XIOLD)
      RETURN
      END

```

```

C      SUBROUTINE AUTO2(RES,DZ)
      SOLVES LINEAR EQUATIONS FOR COEFFICIENTS OF STREAM FUNCTION PSI
      REAL MACH
      COMPLEX ETJ,DZ,ROTATE
      COMMON /A/ GAMMA,MACH
      COMMON /B/ FAREAL,FAIMG,SAREAL,SAIMG,FBIMG,SBIMG,XBIMG,YBIMG
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KB,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /E/ ETJ(64),BUMP(5,10),NBMP
      COMMON /F/ UOLD,VOLD,SINGF,SINGS,SINGX,SINGY,PHIOLD(20),PSIOLD(20)
1    , XOLD(20),YOLD(20),XAIMG,YAIMG,THLAST,CL,DF
      COMMON /G/ N1,N3,N4,N7,M1
      COMMON PHS(129,3),RHX(129,3),RHY(129,3),RHS(129,3),U(129),V(129)
      COMMON ARRAY(128,128)
      NT = NK+1
      DW = (PI+PI)/FLOAT(NF)
      NOSE = NF/(NP+NP)
      OM = -FLOAT(NOSE-1)*DW
C      READ IN PSI
      DO 25 LX = 1,2
      CALL READX (2,MAXO(1,NK/NF))

```

```

FAC = SQRT(2.*(1.-MACH*MACH)/FLOAT(NF))
ROOT2 = 1./SQRT(2.)
CONST = 1./SQRT(FLOAT(NF))
W = OM-PI
DO 20 K = 1,NF
DO 10 J = 2,NF
10 ARRAY(K,J) = FAC*ARRAY(K,J)
W = W+DW
DO 15 J = 1,3
15 RHS(K,J) = RHS(K,J)*FAC
ARRAY(K,NF) = ARRAY(K,NF)*ROOT2
20 ARRAY(K,1) = CONST
25 CONTINUE
CALL LEQ(ARRAY,RHS,NF,3,128,129,SX)
CALL READX (1,MAXO(1,NF/NK))
FAC = FAC*SQRT(FLOAT(NF)/FLOAT(NK))
DO 40 K = 1,NK
DO 30 J = 1,3
30 RHX(K,J) = FAC*RHX(K,J)
DO 35 J = 2,NF
35 ARRAY(K,J) = FAC*ARRAY(K,J)
40 ARRAY(K,NF) = ARRAY(K,NF)*ROOT2
DO 65 M = 1,3
DO 60 K = 1,NK
SX = -RHX(K,M)
DO 50 J = 2,NF
50 SX = SX+RHS(J,M)*ARRAY(K,J)
60 PHS(K,M) = SX
PHS(NK+1,M) = PHS(1,M)
IF (M.NE.3) GO TO 65
PHS(NK+1,3) = PHS(1,3)+(PI+PI)*FAC
65 CONTINUE
CALL READX (4,MAXO(1,NF/NK))
DO 80 M = 1,3
DO 80 K = 1,NK
IF (M.EQ.1) ARRAY(K,NF) = ARRAY(K,NF)*ROOT2
SX = -RHX(K,M)
DO 70 J = 2,NF
70 SX = SX+RHS(J,M)*ARRAY(K,J)
IF (K.EQ.1) RHY(NT,M) = FAC*(SX-RHX(NT,M)+RHX(1,M))
80 RHY(K,M) = FAC*SX
CALL READX (3,MAXO(1,NF/NK))
DO 100 M = 1,3
DO 100 K = 1,NK
IF (M.EQ.1) ARRAY(K,NF) = ARRAY(K,NF)*ROOT2
SX = -RHX(K,M)
DO 90 J = 2,NF
90 SX = SX+RHS(J,M)*ARRAY(K,J)
IF (K.EQ.1) RHX(NT,M) = FAC*(SX-RHX(NT,M)+RHX(1,M))
100 RHX(K,M) = FAC*SX
CALL GETABC(AF,BF,CF,FAC,RES)
DO 110 K = 1,NF
RHS(K,1) = AF*RHS(K,1)+BF*RHS(K,2)+CF*RHS(K,3)
110 CONTINUE
NFH = NF/2
DO 120 J = 1,NFH
120 ETJ(J) = CMPLX(RHS(2*J+1,1),RHS(2*J,1))

```



```

      ETJ(NFH) = CMPLX(0.,RHS(NF,1)*ROOT2)
      PSIOLD(2) = RHS(1,1)/SQRT(2.*(1.-MACH*MACH))
;    GET DX AND DY AT TRAILING EDGE
      MODE = -10
      CALL MAIN (-1,1)
      MODE = 1
      ANGROT = ATAN2(-XBIMG,-YBIMG)
      ROTATE = CMPLX(COS(ANGROT),SIN(ANGROT))
      WRITE (N1) MACH,DF,CL,XAIMG,XBIMG,YAIMG,YBIMG,ROTATE
      DZ = -(PI+PI)* CMPLX(XAIMG+XBIMG,YAIMG+YBIMG)*ROTATE
      RETURN
      END

```

```

      SUBROUTINE READX(IND,ISKP)
;    RESULTS OF INTEGRATION COLLECTED IN ARRAY FOR USE IN AUTO2
;    IND = 1 FOR PHI, IND=2 FOR PSI, IND=3 FOR X, IND=4 FOR Y
;    ONLY EVERY <ISKP>TH POINT IS SAVED
      COMMON PHS(129,3),RHX(129,3),RHY(129,3),RHS(129,3),U(129),V(129)
      COMMON ARRAY(128,128)
      DIMENSION DATA(4,20),SING(4),BPER(4,3)
      DIMENSION DODD(4,128),SING1(4,10)
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /G/ N1,N3,N4,N7,M1
      IF (M1.NE.N1) REWIND M1
      REWIND N1
;    SKIP PAST FIRST FOUR RECORDS ON TAPE1
      READ (M1)
      READ (M1)
      READ (M1)
      READ (M1)
      KX = 0
      NOSE = NK/(NP+NP)
      KSUP=0
      DO 160 L = 1,NP
      MX = 0
      DO 150 JJ = 2,NF,KBM
      M = JJ-1
      MX = MX+1
      NX = MIN0(KBM,NF-M)
      READ (N1) SING
      IF (MX.GT.3) GO TO 20
      DO 10 IX = 1,4
10    BPER(IX,MX) = SING(IX)
20    READ (N1) K1,K2
      IF (M1.EQ.N1) GO TO 30
      READ (M1) SING
      READ (M1) K1,K2
30    INC = -1
      K = KX+(K2-1)/ISKP+1
      IF (K1.GT.0) GO TO 40
      INC = 1
      K = KX

```

```

40 DO 150 LL = 1,K2
C   SKIP PAST FIRST POINT FOR EACH SUBSONIC PATH
   IF (LL.EQ.1) GO TO 90
C   SKIP PAST LAST POINT OF EACH PATH FOR SUPERSONIC PATHS
   IF ((LL.EQ.K2).AND.(K1.GT.0)) GO TO 90
   IF (MOD(LL-1,ISKP).NE.0) GO TO 135
   K = K+INC
   READ (N1)      UK,VK,((DATA(IX,J),IX=1,4),J=1,NX),(SING(IX),IX=1,4)
   IF (MX.GT.3) GO TO 60
   U(K) = UK
   V(K) = VK
   RHX(K,MX) = SING(IND)
60 DO 70 J = 1,NX
70 ARRAY(K,J+M) = DATA(IND,J)
   IF (N1.EQ.M1) GO TO 81
   READ (M1)      UK,VK,((DATA(IX,J),IX=1,4),J=1,NX),(SING(IX),IX=1,4)
C   DO RICHARDSON EXTRAPOLATION
   DO 80 J = 1,NX
80 ARRAY(K,J+M) = (4.*ARRAY(K,J+M)-DATA(IND,J))/3.
   IF (MX.GT.3) GO TO 81
   RHX(K,MX) = (4.*RHX(K,MX)-SING(IND))/3.
81 CONTINUE
   IF ((L.EQ.NP).AND.(LL.EQ.1).AND.(K1.GT.0)) GO TO 82
   IF ((L.EQ.NP).AND.(LL.EQ.K2).AND.(K1.LE.0)) GO TO 82
   GO TO 150
82 IF ((IND.GT.2).AND.(KSUP.GT.0)) GO TO 132
   GO TO 150
90 IF ((L.NE.2).OR.(IND.LE.2)) GO TO 135
C   CORRECT THE ITERATION FOR X AND Y NEAR THE STAGNATION POINT
   READ (N1)      UK,VK,((DATA(IX,J),IX=1,4),J=1,NX),(SING(IX),IX=1,4)
   IF (M1.EQ.N1) GO TO 110
   READ (M1) D,D,(D,D,DATA(1,J),DATA(2,J),J=1,NX),D,D,SING(1),SING(2)
C   DO RICHARDSON EXTRAPOLATION
   DO 100 J = 1,NX
100 DATA(IND,J) = (4.*DATA(IND,J)-DATA(IND-2,J))/3.
   IF (MX.GT.3) GO TO 110
   SING(IND) = (4.*SING(IND)-SING(IND-2))/3.
110 DO 120 J = 1,NX
   JX = J+M
   COR = DATA(IND,J) - ARRAY(KX,JX)
   ARRAY(NOSE,JX) = ARRAY(NOSE,JX) - .5*COR
   DO 120 KZ = NOSE,KX
120 ARRAY(KZ,JX) = ARRAY(KZ,JX) + COR
   IF (MX.GT.3) GO TO 150
   COR = SING(IND) - RHX(KX,MX)
   RHX(NOSE,MX) = RHX(NOSE,MX) - .5*COR
   DO 130 KZ = NOSE,KX
130 RHX(KZ,MX) = RHX(KZ,MX) + COR
   GO TO 150
132 DO 133 J=1,NX
   JX=J+M
   COR=ARRAY(K,JX)-DODO(IND,JX)
   ARRAY(NOSE,JX)=ARRAY(NOSE,JX)-.5*COR
   DO 133 KZ=1,NOSE
   ARRAY(KZ,JX)=ARRAY(KZ,JX)+COR
133 CONTINUE
   IF (MX.GT.3) GO TO 150

```

```

COR=RHX(K,MX)+(PI+PI)*BPER(IND,MX) -SING1(IND,MX)
RHX(NOSE,MX)=RHX(NOSE,MX)-.5*COR
DO 134 KZ=1,NOSE
134 RHX(KZ,MX)=RHX(KZ,MX)+COR
GO TO 150
135 IF((L.NE.1).OR.(IND.LE.2)) GO TO 141
IF(K1.LE.0) GO TO 141
KSUP=K1
READ(N1) UK,VK,((DODO(IX,J+M),IX=1,4),J=1,NX),(SING1(IX,MX),IX=1,4
1)
IF(M1.EQ.N1) GO TO 150
READ(M1) D,D,(D,D,DATA(1,J),DATA(2,J),J=1,NX),D,D,SING(1),SING(2)
DO 136 J=1,NX
JX=J+M
136 DODO(IND,JX)=(4.*DODO(IND,JX)-DATA(IND-2,J))/3.
IF(MX.GT.3)GO TO 150
SING1(IND,MX)=(4.*SING1(IND,MX)-SING(IND-2))/3.
GO TO 150
141 READ(N1)
IF (M1.NE.N1) READ (M1)
150 CONTINUE
160 KX = KX+(K2-1)/ISKP
IF (IND.NE.2) GO TO 180
DO 170 M = 1,3
DO 170 K = 1,NK
170 RHS(K,M) = RHX(K,M)
RETURN
180 DO 190 M = 1,3
190 RHX(NK+1,M) = RHX(1,M)-(PI+PI)*BPER(IND,M)
RETURN
END

```

```

C SUBROUTINE GETABC (AF,BF,CF,CX,DMAX)
SETS UP SINGULARITIES FOR FLOW AT INFINITY AND ADJUSTS COORDINATES
COMPLEX SOFXI,XI, SX,XIBODY
COMMON /B/ FAREAL,FAIMG,SAREAL,SAIMG,FBIMG,SBIMG,XBIMG,YBIMG
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMMON /F/ UOLD,VOLD,SINGF,SINGS,SINGX,SINGY,PHIOLD(20),PSIOLD(20)
1 , XOLD(20),YOLD(20),XAIMG,YAIMG,THLAST,CL,DF
COMMON /G/ N1,N3,N4,N7,M1
COMPLEX XITAIL,BP,XIA,XIB,XIC
COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
COMMON PHS(129,3),RHX(129,3),RHY(129,3),RHS(129,3),U(129),V(129)
COMMON AX(130),BX(130),SS(130),QLG(130),SI(300),QI(300),ZERO(300),
1DQDS(300),DPHDS(300),FP(300),FPP(300),FPPP(300),QPP(300),QPPP(300)
2 , ES(600),Q(600),PHI(600),PHI(130)
COMMON OM(130),DPHI(130,3),PHIPP(130,3),PHIP3(130,3),PHINT(130),
1 DPHII(130),PHIIPP(130),PHIIP3(130) ,DX(130),DY(130),PHIX(130),
2 STAG(130),XIBODY(200),UBODY(200),VBODY(200),XBODY(200),YBODY(200)
DIMENSION DPDO(3),PHIWO(3),PHIWT(3)
DATA RELAX /.8/
N1 = M1

```

```

REWIND N1
READ (N1)
READ (N1) NPTS, SO, PHMN, GAM, (ES(J), Q(J), PHI(J), J = 1, NPTS)
READ (N1) QR, QS, RATC, ITER, NT, (PHII(J), QI(J), J = 1, NT)
READ (N1) WTAIL
C   CONSTRUCT ARRAY OF EVENLY SPACED POINTS AROUND THE CIRCLE
DW = (PI+PI)/FLOAT(NK)
NOSE = NK/(NP+NP)
OM(1) = -FLOAT(NOSE-1)*DW-PI
DO 10 K = 1, NK
OM(K+1) = OM(K)+DW
ZERO(K) = 0.
10 CONTINUE
C   STAGNATION POINT IS AT OM = -PI
DO 20 M = 1, 3
C   SPLINE FIT EACH OF THE SINGULAR SOLUTIONS
CALL PSPLIF(NT, OM, PHS(1, M), DPHI(1, M), PHIPP(1, M), PHIP3(1, M), PHINT)
C   EVALUATE PHI AT -PI
PHIWO(M) = PHS(NOSE, M)
C   EVALUATE DPHI AT -PI
DPDWO(M) = (PHS(NOSE+1, M) - PHS(NOSE-1, M)) / (DW+DW)
20 CONTINUE
CF = GAM / (PHS(NT, 3) - PHS(1, 3))
WELL = GAM - PHMN
C   WT IS CIRCLE ANGLE CORRESPONDING TO THE TRAILING EDGE
WT = WTAIL
C   THE FIRST EQUATION REQUIRES DPHI/DW=0 AT STAGNATION
A11 = DPDWO(1)
A12 = DPDWO(2)
B1 = -CF*DPDWO(3)
C   ITERATE TO FIND WT AT MOST 20 TIMES
DO 80 J = 1, 20
DO 40 M = 1, 3
C   EVALUATE PHI(WT) FOR EACH SINGULAR SOLUTION
CALL INTPL(1, WT, PHIWT(M), OM, PHS(1, M), DPHI(1, M), PHIPP(1, M),
1 PHIP3(1, M))
40 CONTINUE
C   THE SECOND EQUATION FIXES PHI TRAILING EDGE- PHI STAGNATION
A21 = PHIWT(1) - PHIWO(1)
A22 = PHIWT(2) - PHIWO(2)
B2 = -CF * (PHIWT(3) - PHIWO(3)) + WELL
BF = (A11*B2 - A21*B1) / (A11*A22 - A21*A12)
AF = (A22*B1 - A12*B2) / (A11*A22 - A21*A12)
DO 50 K = 1, NT
PHIX(K) = AF*PHS(K, 1) + BF*PHS(K, 2) + CF*PHS(K, 3)
DPHI(K) = AF*DPHI(K, 1) + BF*DPHI(K, 2) + CF*DPHI(K, 3)
PHIIPP(K) = AF*PHIPP(K, 1) + BF*PHIPP(K, 2) + CF*PHIPP(K, 3)
PHIIP3(K) = AF*PHIP3(K, 1) + BF*PHIP3(K, 2) + CF*PHIP3(K, 3)
50 CONTINUE
C   FIND THE LOCATION OF THE TRAILING EDGE
NSTAG = -NK
CALL INTPL(NSTAG, STAG, ZERO, NT, OM, DPHI, PHIIPP, PHIIP3, ZERO)
WTOLD = WT
WT = STAG(1)
IF (NSTAG.LE.1) GO TO 70
DO 60 LL = 2, NSTAG
IF (ABS(STAG(LL) - WTOLD).LT.ABS(WT - WTOLD)) WT = STAG(LL)

```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```

60 CONTINUE
70 IF (ABS(WT-WTOLD).LT.TOL) GO TO 85
80 CONTINUE
  WRITE (N4,200) WT
85 DMAX = 0.
  PHINOS = PHIX(NOSE)
  RFAC = RELAX-1.
  KM = 1
  U(NT) = U(1)
  V(NT) = V(1)
  SX = SOFXI(-CMPLX(COS(TOL),SIN(TOL)))
  U(NOSE) = REAL(SX)
  V(NOSE) = -AIMAG(SX)
  KX = (WT-OM(1))/DW + 1.-TOL
  DELX = -.5*(AF*(RHX(NT,1)-RHX(1,1))+BF*(RHX(NT,2)-RHX(1,2))+
1 CF*(RHX(NT,3)-RHX(1,3)))
  DELY = -.5*(AF*(RHY(NT,1)-RHY(1,1))+BF*(RHY(NT,2)-RHY(1,2))+
1 CF*(RHY(NT,3)-RHY(1,3)))
  RAT = (OM(KX+1)-WT)/DW
  XI = CMPLX(COS(WT),SIN(WT))
  XIBODY(1) = XI
C  GET U AND V AT THE TRAILING EDGE BY LINEAR INTERPOLATION
  UBODY(1) = RAT*U(KX) + (1.-RAT)*U(KX+1)
  VBODY(1) = RAT*V(KX) + (1.-RAT)*V(KX+1)
  DO 110 K = 1,NT
    J = K+KX
    IF (J.GT.NK) J = J-NK
    IF (J.NE.1) GO TO 90
    DELX = -DELX
    DELY = -DELY
90  XBODY(K+1) = AF*RHX(J,1)+BF*RHX(J,2)+CF*RHX(J,3)+DELX
    YBODY(K+1) = AF*RHY(J,1)+BF*RHY(J,2)+CF*RHY(J,3)+DELY
    XI = CMPLX(COS(OM(J)),SIN(OM(J)))
    XIBODY(K+1) = XI
    VBODY(K+1) = V(J)
    UBODY(K+1) = U(J)
    J = K+NOSE-1
    IF (J.GT.NK) J = J-NK
    PHI(K) = PHIX(J)-PHINOS
    IF (J.EQ.NK) PHINOS = PHINOS-GAM
    AX(K) = PHI(K)-PHI(K)
    PHII(K) = PHI(K)+RFAC*AX(K)
    IF (K.LT.NK/2) PHII(K) = AMAX1(PHII(K),PHI(KM))
    KM = K
    IF (ABS(AX(K)).LT.DMAX) GO TO 110
    KMAX = K
    DMAX = ABS(AX(K))
110 CONTINUE
    XIBODY(NT+1) = XIBODY(1)
    UBODY(NK+2) = UBODY(1)
    VBODY(NK+2) = VBODY(1)
C  GET X AND Y AT THE TRAILING EDGE BY FOUR POINT INTERPOLATION
    B1 = .5*(1.+RAT)*(2.-RAT)
    B2 = RAT*(RAT-1.)/6.
    XX = RAT*(XBODY(NK+1)-DELX)+(1.-RAT)*(XBODY(2)+DELX)
    YY = (2.-RAT)*(XBODY(NK)-DELX)+(1.+RAT)*(XBODY(3)+DELX)
    XX = B1*XX+B2*YY

```

```

XBODY(1) = XX-DELX
XBODY(NK+2) = XX+DELX
XX = RAT*(YBODY(NK+1)-DELY)+(1.-RAT)*(YBODY(2)+DELY)
YY = (2.-RAT)*(YBODY(NK)-DELY)+(1.+RAT)*(YBODY(3)+DELY)
YY = B1*XX+B2*YY
YBODY(1) = YY-DELY
YBODY(NK+2) = YY+DELY
DMAX = AX(KMAX).
WTAI = WT
PHI(NT) = GAM
C FIND MINIMUM VALUE OF X
XMIN = XBODY(1)
DO 120 K = 2,NT
120 XMIN = AMIN1(XMIN,XBODY(K))
NTP = NT+1
XMAX = .5*(XBODY(1)+XBODY(NTP))
SCALE = 1./(XMAX-XMIN)
NOSE = NOSE-KX
IF (NOSE.LT.1) NOSE = NOSE+NK
XNOSE = XBODY(NOSE)
YNOSE = YBODY(NOSE)
SF = CX*SCALE
XOLD(2) = SCALE*(DELX-XNOSE)
YOLD(2) = SCALE*(DELY-YNOSE)
CL = SCALE*(CF+CF)*(PHS(NK+1,3)-PHS(1,3))
DO 130 K = 1,NTP
XBODY(K) = SCALE*(XBODY(K)-XNOSE)
130 YBODY(K) = SCALE*(YBODY(K)-YNOSE)
XITAIL = XIBODY(1)
AF = AF*SF
BF = BF*SF
CF = CF*SF
DIS = 1.
IF (NB.NE.1) DIS = -CABS(XIB-XIA)
FAREAL = -AF/DIS
FAIMG = -BF/DIS
FBIMG = -CF
IF (NB.NE.1) FBIMG = FBIMG-FAIMG
REWIND N1
READ (N1)
READ (N1)
WRITE (N1) QR, QS, RATC, ITER, NT, (PHI(J), QI(J) , J = 1, NT)
WRITE (N1) WTAI, (XIBODY(J), UBODY(J), VBODY(J), XBODY(J), YBODY(J),
1 J = 1, NTP), NBPS, (BP(J), J = 1, NBPS)
RETURN
200 FORMAT(* WT DID NOT CONVERGE,--WT=--,E20.10)
END

SUBROUTINE MAIN(ITYP,IPASS)
C ORGANIZES INTEGRATION OF DIFFERENTIAL EQUATIONS ALONG PATHS IN THE
C COMPLEX PLANE
COMPLEX U,V,F1,F2,F3,S1,S2,S3,LAMDAP,LAMDAM,XI,ETA,TAU,PHI,PSI
COMMON PHI(585,1),SKA(390),PSI(585,1),SKB(390),XI(585),ETA(585),

```

```

1 U(585),V(585),F1(585),S1(585),F2(585),S2(585),F3(585),S3(585),
2 LAMDAP(585),LAMDAM(585),TAU(585)
  COMPLEX I,ROOT,ROOT2,X1,Y1,SIG,SIGP,USIG,VSIG,RHDSIG,SOFXI,SOFXIP
  REAL MACH
  COMMON /A/ GAMMA,MACH,ROOT
  COMMON /B/ FAREAL,FAIMG,SAREAL,SAIMG,FBIMG,SBIMG,XBIMG,YBIMG
  COMMON /C/ PI,GRID,TOL
  COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
  COMMON /F/ UOLD,VOLD,SINGF,SINGS,SINGX,SINGY,PHIOLD(20),PSIOLD(20)
1 , XOLD(20),YOLD(20),XAIMG,YAIMG,THLAST,CL,DF
  COMMON /G/ N1,N3,N4,N7,M1
  DATA I/(0.,1.)/ , SJUMP /0./
  IF ((MODE.GT.1).OR.(IPASS.GT.2).OR.(MODE.EQ.-10)) GO TO 10
  IF (MODE.LT.0) THOLD=AIMAG(CLOG(XI(NC)-XI(NB)))
  IR = (PI+PI)/GRID +1.-TOL
  IR = MRP*(1+IR/MAX0(NF,NK))
  IF (MODE.LT.0) IR = MRP
  DIS =-CABS(XI(NB)-XI(1))
  UB = U(NB)
  VB = V(NB)
  QSB = UB*UB+VB*VB
  RHOB = RHO(QSB)
  UA = REAL(U(1))
  VA = REAL(V(1))
  QS = UA*UA+VA*VA
  GAM = (GAMMA-1.)/2.
  CS = 1./(MACH*MACH)+GAM-GAM*QS
  RHOA = RHO(QS)
  QSA = QS
  IF (NB.NE.1) GO TO 10
  XBIMG = 0.
  YBIMG = -TOL
  SIG = SOFXI(XI(1))
  DIS = 1.
  CALL GETHSQ(CMPLX(QS,0.),USIG,VSIG)
  USIG = CONJG(SIG)/(2.*REAL(VSIG)*CMPLX(UA,VA))
  VSIG = .5*I*CMPLX(UA,VA)/CONJG(SIG)
  RHDSIG = -(UA*USIG+VA*VSIG)/CS
  SIGP = SOFXIP(XI(1))
C  READ IN THE NEXT PATH
10 IF (ITYP.GE.0) CALL GTPATH(XI,NC+1,NN,1)
C  GET U, V, F1, S1, LAMDA+, LAMDA-, TAU AND ETA ON XI=CONJG(XIA)
  U(1) = UA
  V(1) = VA
  IF (IPASS.GT.4) GO TO 70
  NJ = -NJ
  CALL LAMBDA(U,V,LAMDAP,LAMDAM,TAU, 1)
  SBIMG = 0.
  GO TO (20,30,40,50) IPASS
20 F1(1) = CMPLX(FAREAL,FAIMG)
  GO TO 60
30 F1(1) = 1./DIS
  FBIMG = 0.
  THOLD = THLAST
  GO TO 60
40 F1(1) = CMPLX(0.,1./DIS)
  FBIMG = 0.

```

```

GO TO 60
50 F1(1) = 0.
   FBIMG = 1.
60 FAREAL = REAL(F1(1))
   S1(1) = F1(1)/TAU(1)
   SAREAL = REAL(S1(1))
   FAIMG = AIMAG(F1(1))
   SAIMG = AIMAG(S1(1))
   QS = UA*UA+VA*VA
   X1 = (UA*F1(1)-VA*(S1(1)/RHOA))/QS
   Y1 = (UA*(S1(1)/RHOA)+VA*F1(1))/QS
   XAIMG = AIMAG(X1)
   YAIMG = AIMAG(Y1)
   F1(1) = FAREAL
   S1(1) = SAREAL
   THLAST = THOLD
   IF (NB.EQ.1) GO TO 70
   SBIMG = -SAIMG
   IF (MODE.GT.0) FBIMG = FBIMG-FAIMG
   QS = UB*UB+VB*VB
   XBIMG = (UB*FBIMG- VB*(SBIMG/RHOB))/QS
   YBIMG = (UB*(SBIMG/RHOB)+VB*FBIMG)/QS
   IF (MODE.GT.0) GO TO 70
   X1 = CMPLX(UA, VA)*CMPLX(YBIMG,XBIMG)
   Y1 = CMPLX(UB, VB)*CMPLX(YBIMG,XBIMG)
   TURN = AIMAG(X1)-AIMAG(Y1)
   DF = 1.-SQRT(QSA/QSB) + PI*TURN/SQRT(QSB)
70 IF (MODE.GT.-10) CALL GETFS0(2,NN)
   IF (NB.NE.1) GO TO 80
   IF (MODE.GT.-10) CALL GETFS2(2,NN)
   XX = FBIMG + AIMAG(SIGP*(USIG*X1+VSIG*Y1))
   YY = AIMAG (SIGP*((USIG+UA*RHOSIG)*Y1-(VSIG+VA*RHOSIG)*X1))
   XAIMG = (UA*XX-VA*YY)/QS
   YAIMG = (UA*YY+VA*XX)/QS
80 IF (MODE.LE.0) II = JJ
   FJUMP = FAIMG+FBIMG
   XJUMP = XAIMG+XBIMG
   YJUMP = YAIMG+YBIMG
   IF (MODE.LT.-9) RETURN
   IF (MODE.GT.0) WRITE (N1) FJUMP,SJUMP,XJUMP,YJUMP
   LIM = MAX0(NX,MODE+1)
   DO 90 L = 1,LIM
   XOLD(L) = 0.
90 YOLD(L) = 0.
   SINGX = XOLD(2)
SINGY = YOLD(2)
C CHECK FOR SUPERSONIC PATH
  IF (KK.GT.0) GO TO 100
C SUBSONIC PATH, COMPUTE SOLUTION IN THE TRIANGLE
  CALL TRIANG(2,NN)
  RETURN
C SUPERSONIC PATH, READ IN THE OTHER PATH
100 CALL GTPATH(XI,NC+1,MM,1)
C SOLVE IN THE RECTANGLE XI(2)≤XI≤XI(JJ) , ETA(2)≤ETA≤ETA(NN)
  DO 110 J = 2,JJ
C GET U,V,F1,S1,LAMDA+,LAMDA-,TAU ON XI=XIA
  CALL GETFS1(J)

```



```

C      KEEP TRACK OF THE BRANCH OF THE SQUARE ROOT FOR LAMDA+,LAMDA-
      ROOT2 = ROOT
      CALL LINEJ(J,2,NN)
      IF (J.GE.NC) CALL GETPSI(NC,J)
      IF (J.EQ.NB) CALL GETFS2(NB+1,NN)
      ROOT = ROOT2
110  CONTINUE
      J = JJ
      L = NN
      M = 1 + (MM-JJ)/IR
      LN = II
      IF (MODE.GE.0) LN = NN
      NCP = NC+1
      DO 120 K = NCP,LN
120  CALL GETPSI(K,J)
      IF (MODE.NE.0) WRITE (N1)      KK,M
      GO TO 140
130  L = L-1
      J = J+1
      CALL GETFS1(J)
      ROOT2 = ROOT
      CALL LINEJ(J,2,L)
      ROOT = ROOT2
      IF (MODE.GE.0) LN = L
      CALL GETPSI(LN,J)
140  IF (MODE.LT.0) GO TO 150
      THSUP = THLAST
      IF (MOD(J-JJ,IR).NE.0) GO TO 160
      WRITE (N1)      UOLD,VOLD,(PHIOLD(K),PSIOLD(K),XOLD(K),YOLD(K),
1  K = 1,NX),SINGF,SINGS,SINGX,SINGY
      GO TO 155
150  IF (MOD(J-JJ,IR).NE.0) GO TO 160
      WRITE (N1) UOLD,VOLD,SINGX,SINGY,SINGF,SINGS,XI(J),ETA(II)
      IF (L.EQ.II) GO TO 160
      XX = SINGX
      YY = SINGY
      CALL SAVE (II+1,L,J)
      SINGX = XX
      SINGY = YY
155  M = M-1
160  CONTINUE
      IF (J.EQ.MM) RETURN
      IF ((MODE.LT.0).AND.(MOD(J-JJ,IR).EQ.0)) WRITE (N1) KK,M
      GO TO 130
      END

```

```

SUBROUTINE TRIANG (K1,K2)
C      SOLVES CHARACTERISTIC INITIAL VALUE PROBLEM FOR SUBSONIC FLOW
      COMMON /A/ SKP(2),ROOT
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMPLEX U,V,F1,F2,F3,S1,S2,S3,LAMDAP,LAMDAM,XI,ETA,TAU,PHI,PSI
      COMMON PHI(585,1),SKA(390),PSI(585,1),SKB(390),XI(585),ETA(585),

```

```

1 U(585),V(585),F1(585),S1(585),F2(585),S2(585),F3(585),S3(585),
2 LAMDAP(585),LAMDAM(585),TAU(585)
   COMPLEX TEMP , ROOT
   LOFF = -1
   IF (MODE.LE.0) LOFF = 0
   DO 30 J = K1,K2
C   SET THE BRANCH FOR THE SQUARE ROOT IN LAMDA+ AND LAMDA-
   ROOT = (0.,1.)
C   GET VALUES OF U,V,F1,S1,LAMDA+,LAMDA-,TAU AT XI(J),ETA(J-1) BY
C   REFLECTION OF THESE QUANTITIES AT XI(J-1),ETA(J)
   U(J-1) = CONJG(U(J))
   V(J-1) = CONJG(V(J))
   F1(J-1) = CONJG(F1(J))
   S1(J-1) = CONJG(S1(J))
   TEMP = CONJG(LAMDAP(J))
   LAMDAP(J-1) = CONJG(LAMDAM(J))
   LAMDAM(J-1) = TEMP
   TAU(J-1) = CONJG(-TAU(J))
C   CHECK TO SEE IF WE ARE PAST XI(NB)
   IF (J.LE.NB) GO TO 10
C   GET F2 AND S2 BY REFLECTION
   F2(J-1) = CONJG(F2(J))
   S2(J-1) = CONJG(S2(J))
C   CHECK TO SEE IF WE ARE PAST XI(NC)
10  IF (J.LE.NC) GO TO 20
C   GET F3 AND S3 BY REFLECTION
   F3(J-1) = CONJG(F3(J))
   S3(J-1) = CONJG(S3(J))
   IF (NX.LE.0) GO TO 20
   DO 15 LX = 1,NX
   PHI(J-1,LX) = CONJG(PHI(J,LX))
15  PSI(J-1,LX+LOFF) = CONJG(PSI(J,LX+LOFF))
C   GET SOLUTION AT XI(J) FROM ETA(J) TO ETA(K2)
20  CALL LINEJ(J,J,K2)
   IF (J.GE.NC) CALL SAVE (J,J,J)
   IF (J.EQ.NB) CALL GETFS2(J+1,K2)
30  CONTINUE
   END

SUBROUTINE LINEJ(J1,K1,K2)
C   SOLVES INITIAL VALUE PROBLEM ALONG A COMPLEX CHARACTERISTIC
COMMON/A/ GAMMA,MACH,ROOT
COMPLEX ROOT
COMMON /B/ FAREAL,FAIMG,SAREAL,SAIMG,FBIMG,SBIMG,XBIMG,YBIMG
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMPLEX U,V,F1,F2,F3,S1,S2,S3,LAMDAP,LAMDAM,XI,ETA,TAU,PHI,PSI
COMMON PHI(585,1),SKA(390),PSI(585,1),SKB(390),XI(585),ETA(585),
1 U(585),V(585),F1(585),S1(585),F2(585),S2(585),F3(585),S3(585),
2 LAMDAP(585),LAMDAM(585),TAU(585)
COMPLEX OXI,EF,GE,RH1,RH2,RH3,RH4,LAMBPH,LAMBM,LAMBPH,LAMBMH,U3,V3
1 , I,XIH,ETAH,PHINC(30),TAUL,DY,ETGE,TAUM,TAUP,DTAUI
COMPLEX T1,T2,WS3,W3,LAMF,L23,L13,WS13,W23,W1,W2,WS1,WS2,LAM1,LAM2

```

```

COMMON /WWS/ W2,WS2,LAM2
DATA I /(0.,1.)/
J = J1
LOFF = -1
IF (MODE.LE.0) LOFF = 0
IF ((J.NE.NC).OR.(K1.GT.NC)) GO TO 20
NEX = 1
KZ = MAX(1-MODE,0)*KK
IF (NB.EQ.1) NEX = 2
DO 10 LX = 1,NX
CALL INITFN(ETA(NC),PHINC(LX),LX)
PSI(NC,LX+LOFF) = 0.
PHI(NC,LX) = PHINC(LX)
IF (KZ.GT.0) GO TO 10
PHI(NC,LX) = REAL(PHINC(LX))
10 CONTINUE
F3(NC) = 0.
S3(NC) = 0.
20 DXI = XI(J)-XI(J-1)
XIH = .5*(XI(J)+XI(J-1))
DO 120 L = K1,K2
IF((KK.GT.0).AND.(MODE.LT.0)) GO TO 21
SOLVE FOR U AND V TO FIRST ORDER ACCURACY
RH1 = V(L)-LAMDAP(L)*U(L)
RH2 = V(L-1)-LAMDAM(L-1)*U(L-1)
U3 = (RH1-RH2)/(LAMDAM(L-1)-LAMDAP(L))
V3 = RH1+LAMDAP(L)*U3
: COMPUTE LAMDA+,LAMDA- TO FIRST ORDER ACCURACY AT XI(J),ETA(L)
CALL LAMBDA(U3,V3,LAMBP,LAMBM,TAUL,-1)
: COMPUTE MIDPOINT VALUES OF LAMDA+ AND LAMDA-
LAMBPH = .5*(LAMBP+LAMDAP(L))
LAMBMH = .5*(LAMBM+LAMDAM(L-1))
: SOLVE FOR U AND V TO SECOND ORDER ACCURACY
RH1 = V(L)-LAMBPH*U(L)
RH2 = V(L-1)-LAMBMH*U(L-1)
U(L) = (RH1-RH2)/(LAMBMH-LAMBPH)
V(L) = RH1+LAMBPH*U(L)
: FIND LAMDA+ AND LAMDA- AT XI(J),ETA(L)
CALL LAMBDA(U(L),V(L),LAMDAP(L),LAMDAM(L),TAUL,1)
GO TO 22
21 CONTINUE
WS1=U(L)+CMPLX(-AIMAG(V(L)),REAL(V(L)))
W1=U(L)-CMPLX(-AIMAG(V(L)),REAL(V(L)))
CALL LAMBD(W1,WS1,LAM1,TAU,-1)
T1=LAM2/(W2*W2)
T2=LAM1/(WS1*WS1)
WS3=(-T1*(T2*WS1-W1)-(T1*W2-WS2))/(1.-T1*T2)
W3=T2*(WS3-WS1)+W1
CALL LAMBD(W3,WS3,LAMF,TAUL,-1)
L23=0.5*(LAM2+LAMF)
L13=0.5*(LAM1+LAMF)
WS13=0.5*(WS3+WS1)
W23=0.5*(W3+W2)
T1=L23/(W23*W23)
T2=L13/(WS13*WS13)
T1=(-T1*(T2*WS1-W1)-(T1*W2-WS2))/(1.-T1*T2)
W2=T2*(T1-WS1)+W1

```

```

      WS2=T1
      CALL LAMBD(W2,WS2,LAM2,TAUL,1)
      U(L)=(W2+WS2)/2.
      V(L)=(WS2-W2)/(2.*I)
22  CONTINUE
C   COMPUTE MIDPOINT VALUE FOR TAU
      TAUP = .5*(TAU(L-1)+TAUL)
      TAUM = -.5*(TAU(L)+TAUL)
      TAU(L) = TAUL
      DTAUI = 1./[TAUM-TAUP]
      IF (NJ.GT.0) GO TO 40
C   COMPUTE F1 AND S1 AT XI(J),ETA(L)
      RH1 = F1(L)+TAUM*S1(L)
      RH2 = F1(L-1)+TAUP*S1(L-1)
      S1(L) = (RH1-RH2)*DTAUI
      F1(L) = RH1-TAUM*S1(L)
C   CHECK TO SEE IF WE HAVE REACHED XIB
      IF((J.LE.NB).OR.(L.LT.NB)) GO TO 120
C   COMPUTE F2 AND S2 AT XI(J),ETA(L)
      RH3 = F2(L)+TAUM*S2(L)
      IF (L.GT.NB) GO TO 30
C   COMPUTE F2,S2 ON ETA = XIB,XI = XI(J)
      RH4 = I*(FBIMG+TAU(L)*SBIMG)
      S2(L) = (RH3-RH4)/(TAUM-TAU(L))
      F2(L) = RH3-TAUM*S2(L)
      GO TO 120
30  RH4 = F2(L-1)+TAUP*S2(L-1)
      S2(L) = (RH3-RH4)*DTAUI
      F2(L) = RH3-TAUM*S2(L)
      IF((J.LT.NC).OR.(L.LT.NC)) GO TO 120
C   COMPUTE THE INHOMOGENEUS TERM
      EF = (RH1+I*(FAIMG+ TAUM *SAIMG))/(XI(1)-XIH)**NEX+ (RH3+I*(FBIMG
1 + TAUM*SBIMG))/(XI(NB)-XIH)
      IF (L.NE.NC) GO TO 50
      IF (J.EQ.NC) GO TO 120
      IF (KZ.LE.0) EF = .5*EF
      S3(L) = S3(L)+DXI*EF/TAUM
35  IF (KZ.GT.0) GO TO 120
      IF (J.EQ.NC) GO TO 120
      DO 38 LX = 1,NX
      CALL INITFN(CONJG(XI(J)),DY,LX)
      DY = .5*(PHINC(LX)+CONJG(DY))
      PSI(L,LX+LOFF) = PSI(L,LX+LOFF)-(DY-PHI(L,LX))/TAUM
38  PHI(L,LX) = DY
      GO TO 120
40  IF ((J.LT.NC).OR.(L.LT.NC)) GO TO 120
      IF (L.EQ.NC) GO TO 35
      IF (J.NE.NC) GO TO 100
      GO TO 70
50  GE = (0.,0.)
C   CHECK FOR SUBSONIC PATH
      IF (KZ.GT.0) GO TO 60
C   PATH IS SUBSONIC, GET SYMMETRIC EF AND GE
      EF = .5*EF
      ETAH = .5*(ETA(L)+ETA(L-1))
      GE = .5*((RH2-I*(FAIMG+ TAUP *SAIMG))/(ETA(1)-ETAH)**NEX + (RH4
1 -I*(FBIMG+ TAUP *SBIMG))/(ETA(NB)-ETAH))

```

```

60 IF (J.NE.NC) GO TO 90
:   GET F3,S3 ON INITIAL CHARACTERISTIC XI = XIC
   ETGE = (ETA(L)-ETA(L-1))*GE
   F3(L) = 0.
   S3(L) = S3(L-1)+ETGE/TAUP
:   GET PSI AND PHI ON THE INITIAL CHARACTERISTIC XI=XIC
70 DO 80 LX = 1,NX
   CALL INITFN(ETA(L),PHI(L,LX),LX)
   IF (KZ.LE.0) PHI(L,LX) = .5*(PHI(L,LX)+CONJG(PHINC(LX)))
80 PSI(L,LX+LOFF) = PSI(L-1,LX+LOFF)-(PHI(L,LX)-PHI(L-1,LX))/TAUP
   GO TO 120
:   COMPUTE F3 AND S3 AT XI(J),ETA(L)
90 DY = DXI*EF
   ETGE = (ETA(L)-ETA(L-1))*GE
   RH1 = F3(L)+TAUM*S3(L)+DY
   RH2 = F3(L-1)+TAUP*S3(L-1) +ETGE
   S3(L) = (RH1-RH2)*DTAUI
   F3(L) = RH1-TAUM*S3(L)
100 DO 110 LX = 1,NX
   RH1 = PHI(L,LX)+TAUM*PSI(L,LX+LOFF)
   RH2 = PHI(L-1,LX)+TAUP*PSI(L-1,LX+LOFF)
   PSI(L,LX+LOFF) = (RH1-RH2)*DTAUI
   PHI(L,LX) = RH1-TAUM*PSI(L,LX+LOFF)
110 CONTINUE
120 CONTINUE
   RETURN
   END

SUBROUTINE SAVE (K1,K2,K3)
:   SAVE THE SOLUTION IN THE REAL HODOGRAPH PLANE FROM ETA(K1) TO
:   ETA(K2) ALONG XI(K3) ON TAPE
COMMON /B/ FAREAL,FAIMG,SAREAL,SAIMG,FBIMG,SBIMG,XBIMG,YBIMG
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMMON /F/ UOLD,VOLD,SINGF,SINGS,SINGX,SINGY,PHIOLD(20),PSIOLD(20)
1  , XOLD(20),YOLD(20),XAIMG,YAIMG,THLAST,CL,DF
COMMON /G/ N1,N3,N4,N7,M1
COMPLEX XITAIL,BP,XIA,XIB,XIC
COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
COMPLEX U,V,F1,F2,F3,S1,S2,S3,LAMDAP,LAMDAM,XI,ETA,TAU,PHI,PSI
COMMON PHI(585,1),SKA(390),PSI(585,1),SKB(390),XI(585),ETA(585),
1  U(585),V(585),F1(585),S1(585),F2(585),S2(585),F3(585),S3(585),
2  LAMDAP(585),LAMDAM(585),TAU(585)
REAL LOG1,LOG2
COMPLEX TEMP,ROTATE
COMMON /TT/ ROTATE,THETA,THETAP,DUM(8)
LOGICAL ISW
DATA UOLD/0./,VOLD/0./,UNEW/0./,VNEW/0./
IF (NJ.GT.0) GO TO 40
LOFF = -1
IF (MODE.LE.0) LOFF = 0
TEMP = CLOG ((XI(K3)-XI(1))/(XI(K3)-XI(NB)))
LOG1 = REAL(TEMP)

```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```

IF (K3.NE.NC) GO TO 10
IR = (PI+PI)/GRID +1.-TOL
IR = MRP*(1+IR/MAX0(NF,NK))
M = 1 + (NN-II)/IR
THETA = AIMAG(TEMP)
THETAP = AIMAG(CLOG(XI(NC)-XI(NB)))
ROTATE = CMPLX(COS(THETAP),-SIN(THETAP))
10 ISW = .TRUE.
DTH = THETA-AIMAG(TEMP)
THETA = AIMAG(TEMP)
IF (ABS(DTH).LT.PI) GO TO 20
FJUMP = FAIMG
SJUMP = SAIMG
XJUMP = XAIMG
YJUMP = YAIMG
DTH = (PI+PI)*DTH/ABS(DTH)
ISW = .FALSE.
20 TEMP = CLOG((XI(K3)-XI(NB))*ROTATE)
LOG2 = REAL(TEMP)
IF (K3.NE.JJ) GO TO 30
IF (MODE.EQ.1) THLAST = AIMAG(TEMP)+THETAP
THETAP = AIMAG(TEMP)+THETAP
ROTATE = CMPLX(COS(THETAP),-SIN(THETAP))
TEMP = LOG2
DTH = THETAP-THLAST
IF (ABS(DTH).LT.PI) GO TO 30
C THIS PATH AND THE LAST FORM A CLOSED LOOP ABOUT XIB
C SET ISW TO MAKE AN APPROPRIATE ADJUSTMENT FOR X AND Y INTEGRATION
ISW = .FALSE.
DTH = (PI+PI)*DTH/ABS(DTH)
THETAP = THETAP-DTH
FJUMP = FBIMG
SJUMP = SBIMG
XJUMP = XAIMG+XBIMG
YJUMP = YAIMG+YBIMG
IF (NB.EQ.1) GO TO 30
. FJUMP = FAIMG+FBIMG
SJUMP = SAIMG+SBIMG
30 THETA2 = AIMAG(TEMP)+THETAP
IF (K1.EQ.NN) THLAST = THETA2
IF (NB.EQ.1) TEMP = 1./(XI(K3)-XI(1))
40 DO 140 L = K1,K2
UNEW = REAL(U(L))
VNEW = REAL(V(L))
UH = .5*(UOLD+UNEW)
VH = .5*(VOLD+VNEW)
QS = UH*UH+VH*VH
IF (NJ.GT.0) GO TO 90
SINGFD = SINGF
SINGSD = SING5
IF (NB.EQ.1) GO TO 50
C COMPUTE SINGULAR TERMS FOR CASCADE
SINGF = REAL(F1(L))*LOG1+(REAL(F1(L))+REAL(F2(L)))*LOG2-FAIMG*
1 THETA-(FAIMG+FBIMG)*THETA2+REAL(F3(L))
SING5 = REAL(S1(L))*LOG1+(REAL(S1(L))+REAL(S2(L)))*LOG2-SAIMG*
1 THETA-(SAIMG+SBIMG)*THETA2+REAL(S3(L))
GO TO 60

```

```

C      COMPUTE SINGULAR TERMS FOR AN AIRFOIL
50 SINGF=REAL((F1(L)+CMPLX(0.,FAIMG))*TEMP)+REAL(F2(L))*LOG2-
1 F8IMG*THETA2 + REAL(F3(L))
   SINGSG=REAL((S1(L)+CMPLX(0.,SAIMG))*TEMP)+REAL(S2(L))*LOG2-
1 S8IMG*THETA2 + REAL(S3(L))
60 IF (MODE.GT.0) GO TO 70
   IR = MRP
   SINGF = SINGF+REAL(PHI(L,1))
   SINGSG = SINGSG+REAL(PSI(L,1)) +PSIOLD(2)
70 IF (L.EQ.NC ) GO TO 90
   IF (ISW) GO TO 80
C      MODIFY THE SINGULAR PART OF X AND Y TO ACCOUNT FOR JUMP IN LOG
   SINGFO = SINGFO+DTH*FJUMP
   SINGSO = SINGSO + DTH*SJUMP
   SINGX = SINGX+DTH*XJUMP
   SINGY = SINGY+DTH*YJUMP
80 DPHI = SINGF-SINGFO
   DPSI =(SINGSG-SINGSO)/RHO(QS)
   SINGX      = SINGX      +(UH*DPHI-VH*DPSI)/QS
   SINGY      = SINGY      +(UH*DPSI+VH*DPHI)/QS
90 DO 110 LX = 1,NX
   IF (L.EQ.NC ) GO TO 100
   DPHI = REAL(PHI(L,LX))-PHIOLD(LX)
   DPSI = (REAL(PSI(L,LX+LOFF))-PSIOLD(LX))/RHO(QS)
   XOLD(LX) = XOLD(LX) +(UH*DPHI-VH*DPSI)/QS
   YOLD(LX) = YOLD(LX) +(UH*DPSI+VH*DPHI)/QS
100 PSIOLD(LX) = PSI(L,LX+LOFF)
   PHIOLD(LX) = PHI(L,LX)
110 CONTINUE
   IF ((L.LT.II).OR.(MODE.EQ.0)) GO TO 130
   IF (MOD(L-II,IR).NE.0) GO TO 130
   IF (MODE.LT.0) GO TO 120
   IF (L.EQ.II) WRITE (N1)      KK,M
   WRITE (N1)      UNEW,VNEW,(PHIOLD(J),PSIOLD(J),XOLD(J),YOLD(J),
1 J = 1,NX),SINGF,SINGSG,SINGX,SINGY
   GO TO 130
120 WRITE (N1)      UNEW,VNEW,SINGX,SINGY,SINGF,SINGSG,XI(K3),ETA(L)
130 UOLD = UNEW
140 VOLD = VNEW
   RETURN
   END

```

```

C      FUNCTION RHO(QS)
      COMPUTE DENSITY RHO AS A FUNCTION OF THE SQUARE OF THE SPEED,QS
      REAL MACH
      COMMON /A/ GAMMA,MACH
      DATA EMOLD /0./
      IF (MACH.EQ.EMOLD) GO TO 10
      EMOLD = MACH
      GAM = (GAMMA-1.)/2.
      COSQ =1./ (MACH*MACH) + GAM
      GAMI = 1./ (GAMMA-1.)
      RHOINF = (MACH*MACH)**GAMI
10 CS = COSQ-GAM*QS
   RHO = RHOINF*CS**GAMI

```

RETURN
END

```

SUBROUTINE GETFS0 (K1,K2)
C  COMPUTE COEFFICIENTS OF SINGULAR TERMS
C  GET U,V,F1 AND S1 ON XI=XIA FROM ETA(K1) TO ETA(K2)
COMMON /B/ FAREAL,FAIMG,SAREAL,SAIMG,FBIMG,SBIMG,XBIMG,YBIMG
COMPLEX U,V,F1,F2,F3,S1,S2,S3,LAMDAP,LAMDAM,XI,ETA,TAU,PHI,PSI
COMMON PHI(585,1),SKA(390),PSI(585,1),SKB(390),XI(585),ETA(585),
1 U(585),V(585),F1(585),S1(585),F2(585),S2(585),F3(585),S3(585),
2 LAMDAP(585),LAMDAM(585),TAU(585)
COMPLEX I,S,T,TAUP,RH1,RH2,SOFXI
DATA I/(0.,1.)/
C  COMPUTE S AT XI = XIA
S = SOFXI(XI(1))
CALL LAMBDA(U(K1-1),V(K1-1),LAMDAP(K1-1),LAMDAM(K1-1),TAU(K1-1),1)
DO 100 L = K1,K2
C  COMPUTE T(ETA(L))
T = CONJG(SOFXI(XI(L)))
ETA(L) = CONJG(XI(L))
U(L) = U(L-1)
V(L) = V(L-1)
CALL GETUV(S,T,U(L),V(L))
CALL LAMBDA(U(L),V(L),LAMDAP(L),LAMDAM(L),TAU(L),1)
C  FIND F1 AND S1 ALONG XI=XIA AT ETA(L)
C  COMPUTE TAU+ AT MIDPOINT
TAUP = .5*(TAU(L)+TAU(L-1))
RH1 = F1(L-1)+TAUP*S1(L-1)
RH2 = -I*(FAIMG-TAU(L)*SAIMG)
S1(L) = (RH1-RH2)/(TAUP+TAU(L))
F1(L) = RH1-TAUP*S1(L)
100 CONTINUE
RETURN
END

```

```

SUBROUTINE GETFS1 (K1)
C  COMPUTE COEFFICIENTS OF SINGULAR TERMS
C  GET U,V,LAMDA+,LAMDA-,TAU,F1 AND S1 AT XI(K1),CONJ(XIA) AND STORE
C  THE RESULTS IN THE FIRST ELEMENT OF THE RESPECTIVE ARRAYS
C  THE FIRST ELEMENT OF THE RESPECTIVE ARRAYS WILL CONTAIN THEIR
C  VALUES AT XI(K1-1),CONJ(XIA)
COMMON /B/ FAREAL,FAIMG,SAREAL,SAIMG,FBIMG,SBIMG,XBIMG,YBIMG
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMPLEX U,V,F1,F2,F3,S1,S2,S3,LAMDAP,LAMDAM,XI,ETA,TAU,PHI,PSI
COMMON PHI(585,1),SKA(390),PSI(585,1),SKB(390),XI(585),ETA(585),
1 U(585),V(585),F1(585),S1(585),F2(585),S2(585),F3(585),S3(585),
2 LAMDAP(585),LAMDAM(585),TAU(585)
COMPLEX I,S,T,RH1,RH2,SOFXI,TAUP,TAUM,DTAUDX

```



```

COMPLEX W2,WS2,LAM2
COMMON /WWS/ W2,WS2,LAM2
DATA I /(0.,1.)/
COMPUTE S AT XI(K1)
S = SOFXI(XI(K1))
COMPUTE T AT ETA = CONJG(XIA)
T = CONJG(SOFXI(XI(1)))
CALL GETUV(S,T,U(1),V(1))
CALL LAMBDA(U(1),V(1),LAMDAP(1),LAMDAM(1),TAUP,1)
IF((KK.LE.0).OR.(MODE.GE.0))GO TO 1
WS2=U(1)+CMPLX(-AIMAG(V(1)),REAL(V(1)))
W2=U(1)-CMPLX(-AIMAG(V(1)),REAL(V(1)))
CALL LAMBDA(W2,WS2,LAM2,TAU,-1)
1 CONTINUE
: FIND F1 AND S1 AT ETA = CONJG(XIA) AT XI(K1)
: COMPUTE TAU- AT THE MIDPOINT
TAUM = -.5*(TAU(1)+TAUP)
RH1 = F1(1)+TAUM*S1(1)
RH2 = I*(FAIMG+TAUP*SAIMG)
S1(1) = (RH1-RH2)/(TAUM-TAUP)
F1(1) = RH1-TAUM*S1(1)
: STORE NEW TAU+
TAU(1) = TAUP
IF (NB.NE.1) RETURN
RH1 = F2(1)+TAUM*S2(1)
RH2 = CONJG(DTAUDX(CONJG(U(1)),CONJG(V(1))))*(S1(1)-I*SAIMG)
RH2 = I*(FBIMG+TAUP*SBIMG)+RH2
S2(1) = (RH1-RH2)/(TAUM-TAUP)
F2(1) = RH1-TAUM*S2(1)
RETURN
END

SUBROUTINE GETFS2(K1,K2)
: COMPUTE COEFFICIENTS OF SINGULAR TERMS
: FIND F2 AND S2 ALONG XI = XIB FROM ETA(K1) TO ETA(K2)
COMMON /B/ FAREAL,FAIMG,SAREAL,SAIMG,FBIMG,SBIMG,XBIMG,YBIMG
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMPLEX U,V,F1,F2,F3,S1,S2,S3,LAMDAP,LAMDAM,XI,ETA,TAU,PHI,PSI
COMMON PHI(585,1),SKA(390),PSI(585,1),SKB(390),XI(585),ETA(585),
1 U(585),V(585),F1(585),S1(585),F2(585),S2(585),F3(585),S3(585),
2 LAMDAP(585),LAMDAM(585),TAU(585)
COMPLEX I,RH1,RH2,TAUP,DTAUDX
DATA I /(0.,1.)/
: CHECK TO SEE IF WE ARE AT XIB
IF (K1.GT.(NB+1)) GO TO 5
: FIND F2 AND S2 AT ETA = CONJG(XIB)
RH1 = -1./TAU(NB)
RH2 = 0.
IF (NB.EQ.1) RH2 = DTAUDX(U(1),V(1))*(S1(1)+I*SAIMG)*RH1
F2(NB) = (AIMAG(RH2)-SBIMG-REAL(RH1)*FBIMG)/AIMAG(RH1)
S2(NB) = REAL(RH2)-REAL(RH1*(F2(NB)+I*FBIMG))
5 IF (K2.LT.K1) RETURN

```

```

      DO 10 L = K1,K2
C      COMPUTE TAU+ AT MIDPOINT
      TAUP = .5*(TAU(L)+TAU(L-1))
      RH1 = F2(L-1)+TAUP*S2(L-1)
      RH2 = -I*(FBIMG-TAU(L)*SBIMG)
      IF (NB.EQ.1) RH2 = RH2 + DTAUDX(U(L),V(L))*(S1(L)+I*SAIMG)
      S2(L) = (RH1-RH2)/(TAUP+TAU(L))
      F2(L) = RH1-TAUP*S2(L)
10  CONTINUE
      RETURN
      END

```

```

      SUBROUTINE GETPSI (L,J)
C      KEEP TRACK OF BRANCHES OF LOGARITHMS ON SUPERSONIC PATHS
      REAL MACH
      COMMON /A/ GAMMA,MACH
      COMMON /B/ FAREAL,FAIMG,SAREAL,SAIMG,FBIMG,SBIMG,XBIMG,YBIMG
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /F/ UOLD,VOLD,SINGF,SINGS,SINGX,SINGY,PHIOLD(20),PSIOLD(20)
1   , XOLD(20),YOLD(20),XAIMG,YAIMG,THLAST,CL,DF
      COMMON /G/ N1,N3,N4,N7,M1
      COMPLEX XITAIL,BP,XIA,XIB,XIC
      COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
      COMPLEX U,V,F1,F2,F3,S1,S2,S3,LAMDAP,LAMDAM,XI,ETA,TAU,PHI,PSI
      COMMON PHI(585,1),SKA(390),PSI(585,1),SK8(390),XI(585),ETA(585),
1   U(585),V(585),F1(585),S1(585),F2(585),S2(585),F3(585),S3(585),
2   LAMDAP(585),LAMDAM(585),TAU(585)
      COMPLEX LOG1,LOG2,I,Z,CLOG,CSINGF,CSINGS,DPHI,DPSI
1   , CU,CV,CPHI(30),CPSI(30),CRHD,UH,VH,QS,CEXP,SINGFO,SINGSO
      COMPLEX FJUMP,SJUMP,ROTATE,F1L,F2L,S1L,S2L,ROT2,F3L,F4L,S3L,S4L
      COMMON /TT/ ROTATE,THETA,THETAP,DUM(8)
      DATA I /(0.,1.)/
C      COMPUTE THE COMPLEX LOG OF Z AND CHOOSE THE BRANCH WHOSE ARGUMENT
C      IS CLOSEST TO TH
      IF (J.NE.NC) GO TO 10
      LOFF = -1
      IF (MODE.LE.0) LOFF = 0
      THETA = AIMAG(CLOG((XI(NC)-XI(1))/(XI(NC)-XI(NB))))
      THETAP = AIMAG(CLOG(XI(NC)-XI(NB)))
      ROTATE = CMPLX(COS(THETAP),-SIN(THETAP))
      ALPHA = -THETA
      ALPHAP = -THETAP
      ROT2 = CONJG(ROTATE)
      GAM = (GAMMA-1.)/2.
      COSQ = 1./(MACH*MACH)+GAM
      GAMI = 1./(GAMMA-1.)
      RHOINF = (MACH*MACH)**GAMI
      CU = U(NC)
      CV = V(NC)
      FAC = CTHR
      IF (MODE.LE.0) FAC = 0.
10  UH = .5*(CU+U(L))

```

```

VH = .5*(CV+V(L))
QS = UH*UH+VH*VH
CRHO = RHOINF*CEXP(GAMI*CLOG(COSQ-GAM*QS))
IF (NJ.GT.0) GO TO 130
ISW=0
LOG1 = CLOG ((XI(J) -XI(1))/(XI(J) -XI(NB)))
DTH = THETA-AIMAG(LOG1)
THETA = AIMAG(LOG1)
IF(MODE.LT.0) GO TO 20
IF (ABS(DTH).LT.PI) GO TO 20
WRITE (N4,170)
CALL EXIT
20 IF (NB.EQ.1) LOG1 = 1./(XI(J)-XI(1))
LOG2 = CLOG((XI(J)-XI(NB))*ROTATE) + CMPLX(0.,THETAP)
IF (MODE.EQ.1) THLAST = AIMAG(LOG2)
THETAP = AIMAG(LOG2)
ROTATE = CMPLX(COS(THETAP),-SIN(THETAP))
DTH = THETAP-THLAST
IF(MODE.LT.0) GO TO 30
IF (ABS(DTH).LT.PI) GO TO 30
DTH = (PI+PI)*DTH/ABS(DTH)
THETAP = THETAP-DTH
LOG2 = LOG2-CMPLX(0.,DTH)
ISW=1
XJUMP = XBING+XAING
YJUMP = YBING+YAING
SINGX=SINGX+DTH*XJUMP
SINGY=SINGY+DTH*YJUMP
30 THLAST = THETAP
40 IF (NB.NE.1) LOG1 = LOG1+LOG2
F1L = F1(L)+I*FAING
F2L = F2(L)+I*FBING
S1L = S1(L)+I*SAING
S2L = S2(L)+I*SBING
CSINGF = F1L*LOG1+F2L*LOG2+F3(L)
CSINGS = S1L*LOG1+S2L*LOG2+S3(L)
IF (MODE.LE.0) GO TO 90
IF (NB.EQ.1) GO TO 50
LOG1 = CLOG((ETA(L)-ETA(1))/(ETA(L)-ETA(NB)))
DTH = ALPHA-AIMAG(LOG1)
ALPHA = AIMAG(LOG1)
IF (ABS(DTH).LT.PI) GO TO 60
WRITE(N4,170)
CALL EXIT
50 LOG1 = 1./(ETA(L)-ETA(1))
60 LOG2 = CLOG((ETA(L)-ETA(NB))*ROT2)+CMPLX(0.,ALPHAP)
ALPHAP = AIMAG(LOG2)
ROT2 = CMPLX(COS(ALPHAP),-SIN(ALPHAP))
DTH = ALPHAP+THLAST
IF (ABS(DTH).LT.PI) GO TO 80
DTH = (PI+PI)*DTH/ABS(DTH)
ISW=1-ISW
IF(ISW.EQ.0) GO TO 70
WRITE(N4,160)
CALL EXIT
70 ALPHAP=ALPHAP-DTH
LOG2 = LOG2-CMPLX(0.,DTH)

```

```

80 IF (NB.NE.1) LOG1 = LOG1+LOG2
   F3L = F1(L)-I*FAIMG
   F4L = F2(L)-I*FBIMG
   S3L = S1(L)-I*SAIMG
   S4L = S2(L)-I*S8IMG
   CSINGF = .5*(CSINGF+F3L*LOG1+F4L*LOG2+F3(L))
   CSINGS = .5*(CSINGS+S3L*LOG1+S4L*LOG2+S3(L))
   GO TO 100
90 CSINGF = CSINGF+PHI(L,1)
   CSINGS = CSINGS+PSI(L,1)+PSIOLD(2)
100 IF (J.EQ.NC) GO TO 120
110 DPHI = CSINGF-SINGFO
   DPSI = (CSINGS-SINGSO)/CRHO
   SINGX = SINGX +(UH*DPHI-VH*DPSI)/QS
   SINGY = SINGY +(UH*DPSI+VH*DPHI)/QS
120 SINGFO = CSINGF
   SINGSO = CSINGS
   SINGF = REAL(CSINGF)
   SINGS = REAL(CSINGS)
   SINGS = SINGS+FAC*AIMAG(CSINGS)
130 CU = U(L)
   UOLD = CU
   CV = V(L)
   VOLD = CV
   DO 150 LX = 1,NX
   IF (J.EQ.NC) GO TO 140
   DPHI = PHI(L,LX)-CPHI(LX)
   DPSI = (PSI(L,LX+LOFF)-CPSI(LX))/CRHO
   XOLD(LX)=XOLD(LX) +(UH*DPHI-VH*DPSI)/QS
   YOLD(LX)=YOLD(LX) +(UH*DPSI+VH*DPHI)/QS
140 CPHI(LX) = PHI(L,LX)
   CPSI(LX) = PSI(L,LX+LOFF)
   PSIOLD(LX) = CPSI(LX)
   PSIOLD(LX) = PSIOLD(LX)+FAC*AIMAG(PSI(L,LX+LOFF))
150 PHIOLD(LX) = PHI(L,LX)
   RETURN
160 FORMAT (//4X64H***AUTOMATED SUPERSONIC PATH CROSSES CUT FROM XIC T
170 1THROUGH XIB***)
170 FORMAT(//6X61H****AUTOMATED SUPERSONIC PATH CROSSES CUT FROM XIA T
10 XIB****)
   END

```

C SUBROUTINE GETUV-(S,T,U,V)-
 GIVEN S AND T FIND U AND V
 COMMON /G/ N1,N3,N4,N7,M1
 COMPLEX S,T,U,V, QS,W,WSTR,I,HSQ,HSQPR,QSNEW,ST,CSQRT
 EXTERNAL CSQRT
 DATA I /(0.,1.)/ , TOL /1.E-10/
 QS = U*U+V*V
 W = S
 WSTR = T
 ST = S*T
 IF (CABS(ST).GE.TOL) GO TO 25

```

CALL GETHSQ((0.,0.),HSQ,HSQPR)
W = W/SQRT(REAL(HSQPR))
WSTR = WSTR/SQRT(REAL(HSQPR))
GO TO 100
DO AT MOST 20 NEWTON ITERATIONS
25 DO 80 L = 1,20
  COMPUTE H(QS)**2 AND ITS DERIVATIVE WITH RESPECT TO QS, HSQPR
  CALL GETHSQ(QS,HSQ,HSQPR)
  QSNEW = QS-(HSQ-ST)/HSQPR
  IF (CABS(QSNEW-QS).LT.TOL) GO TO 90
80 QS = QSNEW
  WRITE (N4,110) ST
  QSNEW=U*U+V*V
  QS=QSNEW
90 IF (CABS(ST).LT..1) GO TO 95
  QS = QSNEW
  W = CSQRT(QS*(S/T),W)
  WSTR = W*(T/S)
  GO TO 100
95 W = S*CSQRT(QS/HSQ,(1.,0.))
  WSTR = T*CSQRT(QS/HSQ,(1.,0.))
100 U = .5*(W+WSTR)
  V = .5*I*(W-WSTR)
  RETURN
110 FORMAT(42H NEWTON ITERATION DID NOT CONVERGE AT ST =,2E12.4)
END

```

```

SUBROUTINE GETHSQ(QS,HSQ,HSQPR)
COMPUTE H(QS)**2 AND ITS DERIVATIVE WITH RESPECT TO QS, HSQPR
REAL MACH
COMMON /A/ GAMMA,MACH,RR,ROOT1
COMPLEX QS,HSQ,HSQPR,BPHI1,BPHI2,CSQRT,ROOT2,TEMP,QSX,CS,ROOT1,RR
EXTERNAL CSQRT
DATA EMOLD/0./
IF (MACH.EQ.EMOLD) GO TO 10
CONST = 1.
GAM = (GAMMA-1.)/2.
COSQ = 1./((MACH*MACH) + GAM
QSTRSQ = (2./((GAMMA+1.))*COSQ
EMU = SQRT((GAMMA-1.)/(GAMMA+1.))
FAC = EMU/QSTRSQ
QSX = QS
QS = (1.,0.)
10 CS = COSQ-GAM*QS
BPHI1 = (CS+CS)/QSTRSQ-1.
BPHI2 = (CS+CS)-QS
ROOT2 = CSQRT((CS+CS)*(BPHI2-QS),ROOT1)
ROOT1 = FAC*ROOT2
TEMP = CEXP(CLOG(BPHI1-ROOT1)/EMU)
TEMP = CONST/(TEMP*(BPHI2+ROOT2))
TEMP IS H**2/QS
HSQ = QS*TEMP
IF (MACH.EQ.EMOLD) GO TO 20

```

```

      EMOLD = MACH
C     SET CONST SO THAT H(1) = 1
      CONST = 1./REAL(HSQ)
      QS = QSX
      GO TO 10
C     COMPUTE THE DERIVATIVE OF HSQ WITH RESPECT TO QS
20    HSQPR = TEMP*ROOT2/(CS+CS)
      RETURN
      END

```

```

C     SUBROUTINE LAMBDA(UL,VL,LAMDAP,LAMDAM,TAU,IND)
      COMPUTES LAMBDA+ AND LAMBDA- GIVEN U AND V
      REAL MACH
      COMMON /A/ GAMMA,MACH,ROOT
      COMPLEX UL,VL,LAMDAP,LAMDAM,TAU,U,V,CS,QS,ROOT,CSQRT,CLOG,CEXP
      EXTERNAL CSQRT
      DATA EMOLD/0./
      IF (MACH.EQ.EMOLD) GO TO 10
      EMOLD = MACH
      GAM = (GAMMA-1.)/2.
      COSQ = 1./((MACH*MACH) + GAM
      GAMI = -1./((GAMMA-1.))
      RHOINF = (MACH*MACH)**GAMI
      GAMI = GAMMA*GAMI
10    U = UL
      V = VL
      QS = U*U+V*V
      CS = COSQ - GAM*QS
      ROOT = CSQRT(CS*(QS-CS),ROOT)
      LAMDAP = (U*V+ROOT)/(CS-V*V)
      LAMDAM = (U*V-ROOT)/(CS-V*V)
      IF (IND.LT.0) RETURN
      TAU = RHOINF*CEXP(GAMI*CLOG(CS))*ROOT
      RETURN
      END

```

```

C     SUBROUTINE LAMBD(WL,WSL,LAM,TAU,IND)
      COMPUTES LAM GIVEN W AND WS INSTEAD OF U AND V
      REAL MACH
      COMMON/A/ GAMMA,MACH,ROOT
      COMPLEX WL,WSL,LAM,TAU ,W,WS,CS,QS,ROOT,CSQRT,CLOG,CEXP,I
      EXTERNAL CSQRT
      DATA EMOLD/0./,I/(0.,1.)/
      IF(MACH.EQ.EMOLD) GO TO 10
      EMOLD=MACH
      GAM=(GAMMA-1.)/2.
      COSQ=1./((MACH*MACH)+GAM
      GAMI=-1./((GAMMA-1.))
      RHOINF=(MACH*MACH)**GAMI

```

```

      GAMI=GAMMA*GAMI
10  W=WL
      WS=WSL
      QS=W*WS
      CS=CSQ- GAM*QS
      ROOT=CSQRT(CS*(QS-CS),ROOT)
      LAM=2.*CS-QS-2.*I*ROOT
      IF(IND.LT.0)RETURN
      TAU=RHOINF*CEXP(GAMI*CLOG( CS))*ROOT
      RETURN
      END

```

```

      COMPLEX FUNCTION DTAUDX(UL,VL)
      COMPLEX UL,VL,QS,CS,S,SOFXI,SOFXIP,CEXP,CLOG,DSDXI,CONST
      REAL MACH
      COMMON /A/ GAMMA,MACH,ROOT
      COMPLEX XITAIL,BP,XIA,XIB,XIC
      COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
      DATA EMOLD/0./
      IF (MACH.EQ.EMOLD) GO TO 10
      EMOLD = MACH
      GAM = (GAMMA-1.)/2.
      CSQ = 1./(MACH*MACH) + GAM
      S = SOFXI(XIA)
      DSDXI = SOFXIP(XIA)
      GAMI = -1./(GAMMA-1.)
      RHOINF = (MACH*MACH)**GAMI
      GAMI = GAMMA*GAMI
      CONST = CMPLX( 0.,-.25*(GAMMA+1.)*RHOINF)
10  QS = UL*UL+VL*VL
      CS = CSQ-GAM*QS
      DTAUDX = CONST*CEXP(GAMI*CLOG(CS))*QS*QS*DSDXI/(S*(QS-CS))
      RETURN
      END

```

```

      SUBROUTINE INITFN (ETA,Z,LJ)
      COMPUTES THE INITIAL CHARACTERISTIC DATA ON XI = XIC
      Z IS THE VALUE OF THE INITIAL FUNCTION
      COMPLEX ETA,Z,XIP,I,TEMP,ETJ
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /E/ ETJ(64),BUMP(5,10),NBMP
      DATA I /(0.,1.)/
      LX = LJ+IABS(NJ)
      IF (MODE.LE.0) GO TO 10
      J = (LX+1)/2
      TEMP = 1.
      IF (LX.NE.J+J) TEMP = -I
      Z = TEMP*ETA**J
      RETURN
10  XIP = CONJG(ETA)

```

```

C      SUM UP THE POLYNOMIAL TERMS
      J = NF/2
      Z = ETJ(J)
20     J = J-1
      Z = Z*XIP+ETJ(J)
      IF (J.GT.1) GO TO 20
      Z = Z*XIP
      Z = CONJG(Z)
      RETURN
      END

```

```

C      SUBROUTINE GTPATH (XI,K1,K2,K3)
C      READS IN THE PATHS ON THE COMPLEX CHARACTERISTICS AND SETS UP THE
C      INITIAL GRID
C      PATHS ARE PRESCRIBED ON THE PLANE XI=CONJG(XIA) AND THE
C      GRID IS SET UP IN THE PLANE ETA=XIA
      REAL MACH
      COMMON /A/ GAMMA,MACH
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /G/ N1,N3,N4,N7,M1
      COMPLEX XITAIL,BP,XIA,XIB,XIC
      COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
      REAL MACHA,MACHB,MTAIL
      COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
      COMPLEX POINT,H,S,XIOFS,SOFXI,XI(1)
      DATA EMOLD/0./ , K2MAX/0./ ,SNEW/0./ ,SOLD/0./
      IF (MODE.EQ.99) GO TO 80
      IF (MACH.EQ.EMOLD) GO TO 10
      EMOLD = MACH
      QCRIT = (GAMMA-1.)/(GAMMA+1.) + 2./((GAMMA+1.)*MACH*MACH)
      CALL GETHSQ (CMPLX(QCRIT,0.),H,POINT)
      SCRIT = REAL(H)
      NPH = 1+NP/2
      IX=(PI+PI)/.04+1.-TOL
      PIX = (PI+PI)/FLOAT(NP)
      IR = (NK/NP)*(IX/NK+1)
      DSS= PIX/FLOAT(IR)
      PIX2 = .5*PIX
      IF (MOD(IR,2).NE.0) PIX2 = PIX2+.5*DSS
10     IF (MODE.LT.0) GO TO 40
      IF (K3.EQ.0) GO TO 50
      TH = FLOAT(MODE-NPH)*PIX-PIX2
      IF (KK.GT.0) TH = TH+PIX
      FAC = 1.
      IF (KK.GT.0.) FAC = -1.
      CALL PATH (XI,1.,TH,TH+FAC*PIX,K1,K2)
      KX = 0
      POINT = XI(JJ)
      DO 20 J = JJ,K2
      S = SOFXI(XI(J))
      IF (REAL(S)*REAL(S)+AIMAG(S)*AIMAG(S).LT.SCRIT) GO TO 20
      IF (KX.EQ.0) POINT = XIOFS(CONJG(SCRIT/SOFXI(XI(J))),XI(J))

```



```

      KX = KX+1
20  CONTINUE
      IF (KX.EQ.0) GO TO 30
      THF = ATAN2(AIMAG(POINT),REAL(POINT))
      RADF = CABS(POINT)
      THF=THF-.1*FAC/(RADF**2)
      IF((TH.LT.-PI).AND.(THF.GT.0.)) TH=TH+PI+PI
      IF (FAC*(THF-TH).GT.0.) GO TO 30
21  CONTINUE
      :  MODIFY THE SUPERSONIC PATH TO AVOID SINGULAR POINTS
      CALL PATH(XI,RADF,THF,THF,K1,K2)
      CALL PATH(XI,1.,TH,TH+FAC*PIX,K2+1,K2)
30  KK = KX
      K2MAX = MAX0(K2MAX,K2)
      IF (K2.LE.L8M/3) RETURN
      MSG = 10HAUTOMATION
      WRITE (N4,140) MSG
      CALL EXIT
      :  CONSTRUCT SUPERSONIC PATHS
40  IF (KK.EQ.0) GO TO 50
      CALL PATH(XI,.96,THL+.07,THL+.07,K1,K2)
      CALL PATH(XI,1.04,THL+.04,THF+.04,K2+1,K2)
      CALL PATH(XI,1.,THF,THL,K2+1,K2)
      GO TO 130
50  IF(MODE.LT.-1) L=L-1
      IF(MODE.LT.-1) GO TO 60
      L = 0
      DT = (PI+PI)/FLOAT(IX)
      ANGT=ATAN2(AIMAG(XITAIL),REAL(XITAIL))
      RAT=ANGT/DT
      IF(FLOAT(INT(RAT)).GE.RAT) GO TO 51
      ITEST=INT(RAT)+1
      GO TO 52
51  ITEST=INT(RAT)
52  CONTINUE
      THFO=FLOAT(ITEST)*DT
      POINT=CMPLX(COS(THFO),SIN(THFO))
      THLO=THFO
      ICOUNT = 0
      H = CMPLX(COS(DT),SIN(DT))
      SEARCH FOR SUPERSONIC POINT
60  S = SOFXI(POINT)
      SNEW=TOL
      L = L+1
70  SOLD = SNEW
      SNEW = REAL(S)*REAL(S)+AIMAG(S)*AIMAG(S)
      IF((ICOUNT.GT.0).AND.ABS(SNEW-SCRIT).GT.ABS(SOLD-SCRIT)) GO TO 91
      IF (ICOUNT.GT.0) GO TO 90
      THF=FLOAT(L-1)*DT+THFO
      DTHF=-DT
      IF (SNEW.GT.SCRIT) GO TO 90
      POINT = POINT*H
      IF (L.LT.IX) GO TO 60
      MODE = -10
80  JJ = K2MAX
      K2MAX = 0
      RETURN

```

```

90 DTHF = -DTHF*(SNEW-SCRIT)/(SNEW-SOLD)
   THF = THF-DTHF
   S = SOFXI(CMPLX(COS(THF),SIN(THF)))
   ICOUNT = ICOUNT+1
   IF ((ICOUNT.LT.20).AND.(ABS(DTHF).GT.TOL)) GO TO 70
   GO TO 92
91 THF=THF+DTHF
   WRITE(N4,150)
   SNEW=TOL
92 CONTINUE
   ICOUNT = 0
C   NOW FIND THE NEXT SUBSONIC POINT
100 POINT = POINT*H
   L = L+1
   S = SOFXI(POINT)
110 SOLD = SNEW
   SNEW = REAL(S)*REAL(S)+AIMAG(S)*AIMAG(S)
   IF ((ICOUNT.EQ.0).AND.(SNEW.GT.SCRIT)) GO TO 100
   IF(ICOUNT.GT.0.AND.ABS(SNEW-SCRIT).GT.ABS(SOLD-SCRIT)) GO TO 111
   IF(ICOUNT.EQ.0) THL=FLOAT(L-1)*DT+THL0
   IF(ICOUNT.EQ.0) DTHL=-DT
   DTHL = -DTHL*(SNEW-SCRIT)/(SNEW-SOLD)
   THL = THL-DTHL
   ICOUNT = ICOUNT+1
   S = SOFXI(CMPLX(COS(THL),SIN(THL)))
   IF ((ICOUNT.LT.20).AND.(ABS(DTHL).GT.TOL)) GO TO 110
   GO TO 112
111 THL=THL+DTHL
   WRITE(N4,150)
   SNEW=TOL
112 CONTINUE
   ICOUNT = 0
   IF (K3.NE.0) GO TO 120
   XI(1) = CMPLX(THF,THL)
   RETURN
120 KK = -MODE
   CALL PATH(XI,.96,THF-.07,THF-.07,K1,K2)
   CALL PATH(XI,1.04,THF-.04,THL-.04,K2+1,K2)
   CALL PATH(XI,1.,THL,THF,K2+1,K2)
130 K2MAX = MAX0(K2MAX,K2)
   IF (K2.LE.LBM) RETURN
   MSG = 10HSUPERSONIC
   WRITE (N4,140) MSG
   CALL EXIT
140 FORMAT (///5X5H**** ,A10,23H PATH IS TOO LONG ***** )
150- FORMAT(///12X62H****NEWTON ITERATION FOR SONIC POINT DID NOT CONVE
   1RGE WELL****)
   END

```

```

SUBROUTINE PATH(XI,RAD,TH,THL,K1,K2)
  COMPLEX XI(1),H,POINT,S,SOFXI,XIOFS,ROOT,ROOT1
C  CONSTRUCT POINTS XI(J) ON PATH FOR J = K1 TO K2
C  IF PATH STARTS AT XIC WE START OUR PATH ON A CIRCLE OF RADIUS

```

```

C      ABS(XIC-XIB) CENTERED AROUND XIB
C      A STRAIGHT LINE IS THEN DRAWN TO A POINT RADIUS=RAD AND ARG=TH
REAL MACH
COMMON /A/ GAMMA,MACH,ROOT,ROOT1
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KB,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
DATA EMOLD/0./
IF (MACH.EQ.EMOLD) GO TO 10
EMOLD = MACH
IR = (PI+PI)/GRID + 1.-TOL
IR = (NK/NP)*(IR/NK+1)
PIX = (PI+PI)/FLOAT(NP)
DSS = (PIX+PIX)/FLOAT(IR)
QCRIT = (GAMMA-1.)/(GAMMA+1.)+2./((GAMMA+1.)*MACH*MACH)
CALL GETHSQ(CMPLX(QCRIT,0.),H,POINT)
SCRIT = REAL(H)
10 K = K1-1
KN = K1
LMODE = 0
POINT = RAD*CMPLX(COS(TH),SIN(TH))
IF (K.NE.NC) GO TO 30
ROOT1 = 1.
H = XI(NC)-XI(NB)
CA = CABS(H)
ARGC = ATAN2(AIMAG(H),REAL(H))
H = POINT-XI(NB)
THX = ATAN2(AIMAG(H),REAL(H))
IF (ARGC-THX.GT.PI) ARGC = ARGC-PI-PI
IF (THX-ARGC.GT.PI) ARGC = ARGC+PI+PI
CARC = CA*(THX-ARGC)
MM = ABS(CARC)/GRID +.5
MM = MM*MRP
KN = K1+MM
K = KN-1
IF (MM.EQ.0) GO TO 30
DT = (THX-ARGC)/FLOAT(MM)
H = CMPLX(COS(DT),SIN(DT))
DO 20 J = K1,K
20 XI(J) = XI(NB)+(XI(J-1)-XI(NB))*H
30 H = POINT-XI(K)
ABSHSQ = REAL(H)*REAL(H)+AIMAG(H)*AIMAG(H)
IF (ABSHSQ.LT.1.E-10) GO TO.45
C      FIND THE MINIMUM DISTANCE FROM XIA TO THE PATH
S = POINT-XI(1)
T = (REAL(S)*REAL(H)+AIMAG(H)*AIMAG(S))/ABSHSQ
IF (T.GT.1.) GO TO 35
T = AMAX1(0.,T)
S = POINT-T*H
DIS = CABS(S-XI(1))
IF (DIS.GT.GRID) GO TO 35
LMODE = 1
FAC = 1.2*GRID/DIS
POINT = XI(1)+FAC*(S-XI(1))
GO TO 30
35 MM = SQRT(ABSHSQ)/GRID + 1.-TOL
MM = MM*MRP
JJ = K + MM

```

```

      IF (MM.EQ.0) GO TO 45
      H = H/FLOAT(MM)
      DO 40 J = KN,JJ
40  XI(J) = XI(J-1) + H
45  KN = JJ+1
      IF (LMODE.EQ.0) GO TO 48
      LMODE = 0
      POINT = RAD*CMPLX(COS(TH),SIN(TH))
      K = JJ
      GO TO 30
48  IX = ABS(THL-TH)/DSS+.5
      IX = IX+IX
      K2 = JJ+IX*MRP
      IF (IX.EQ.0) RETURN
      DS = (THL-TH)/FLOAT(IX*MRP)
      H = CMPLX(COS(DS),SIN(DS))
      DO 50 J = KN,K2
50  XI(J) = XI(J-1)*H
      IF((MODE.GT.0).OR.(K1.EQ.NC+1).OR.(RAD.NE.1.))RETURN
C  PASTE POINTS TO THE SONIC LOCUS
60  DO 70 J = KN,K2
      S = SQFXI(XI(J))
      ABSSQ = REAL(S)*REAL(S)+AIMAG(S)*AIMAG(S)
      XI(J) = XIOFS(S*SQRT(SCRIT/ABSSQ),XI(J-1))
70  CONTINUE
      RETURN
      END

C  SUBROUTINE BODYPT(MX,NPTS)
      READS TAPE1 DATA, FINDS BODY POINTS AND CONSTRUCTS P-N DIAGRAMS
      REAL MACH
      COMMON /A/ GAMMA,MACH
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /F/ ROTATE,XAIMG,XBIMG,YAIMG,YBIMG,CL,DF
      COMMON /G/ N1,N3,N4,N7,M1
      COMPLEX XITAIL,BP,XIA,XIB,XIC
      COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
      REAL MACHN,KAPPA
      COMPLEX XI,XIBODY,CHAR,XIS
      COMMON CHAR(61,20),XI(300),XIS(300),XIBODY(300),UBODY(300),
1  VBODY(300),XBODY(300),YBODY(300),H(300),THETA(300),ANGL(300),
2  MACHN(300),KAPPA(300),XREAL(300),YREAL(300),S(300),ES(600),Q(600),
3  , PHI(600),QL(300),ILINE(300)
      COMMON UREAL(300),VREAL(300),PSI(300),FEE(300),SX(300),QX(300)
      COMPLEX ROTATE
      DATA NROW /1/ , MXMAX /300/ , QTOL /0./
      QTOL=TOL
      MRX=2
      IPL = MOD(IABS(IPLT)/10,10)
      ILINE(MX) = 0
      ILINE(MX+1) = 0
      REWIND M1

```

```

REWIND N1
IF (N1.NE.M1) N3 = N7
READ (M1) NRN,NIN,{SX(J),QX(J), J = 1,NIN}
READ (M1) NPTS,DUM,DUM,DUM,{ES(J),Q(J),PHI(J), J = 1,NPTS}
IF (IPL.EQ.0) GO TO 25
C PLOT Q(S)
SF = 10./{SX(NIN)-SX(1)}
CALL SETSF(SF)
CALL CHSIZE(.07)
CALL ORIGIN (6.0,5.5)
SFI = 2.5/SF
CALL CPLOT (CMPLX(ES(1),SFI*Q (1)),3)
DO 10 L = 2,NPTS
10 CALL CPLOT (CMPLX(ES(L),SFI*Q(L)),2)
DO 20 L = 1,NIN
20 CALL CSYMBL (CMPLX(SX(L),SFI*QX(L)),3,-1)
CALL SETSF (1.)
CALL CHSIZE(.14)
CALL XYAXES ((0.,0.),5.,5.,.2,0.,6H(F5.1))
CALL CSYMBL ((4.5,-.6),1HS,1)
CALL XYAXES ((-5.0,0.),4.,4.,.40,90.,6H(F5.1))
CALL CSYMBL ((-4.7,3.7),1HQ,1)
CALL CSYMBL ((-1.5,-4.5),24HINPUT SPEED DISTRIBUTION ,24)
CALL FRAMER
CALL ORIGIN (3.5,2.0)
IF((IPL.EQ.2).OR.(IPL.EQ.4).OR.(IPL.EQ.7).OR.(IPL.EQ.9)) CALL ORIG
IN(3.5,4.0)
IF (CABS(XIA-XIB).GT.TOL) CALL ORIGIN (3.5,1.5)
CALL SETSF (5.)
25 READ (M1) QR,QS,RATC
IF (MX.EQ.1) RETURN
READ (M1) DUM,{XIBODY(J),UBODY(J),VBODY(J),XBODY(J),YBODY(J) ,
1 J = 1,MX) ,NBPS,{BP(J), J = 1,NBPS}
IF (EOF(M1).NE.0) RETURN
READ (M1) MACH,DF,CL,XAIMG,XBIMG,YAIMG,YBIMG,ROTATE
IF (EOF(M1).NE.0) RETURN
COSA=REAL(ROTATE)
SINA=AIMAG(ROTATE)
YR=XBODY(1)*SINA+YBODY(1)*COSA
IF((YR.LT.0.).AND.(NB.GT.1)) CALL CPLOT(CMPLX(0.,-YR),-3)
QCRIT = (GAMMA-1.)/(GAMMA+1.) + 2./((GAMMA+1.)*MACH*MACH)
MX = MX+1
C READ DATA FOR A PATH OR FORK
30 READ (N1) KK,NN
C ****CHECK FOR END OF FILE****
IF (EOF(N1).NE.0) GO TO 160
DO 40 L = 1,NN
READ (N1) UREAL(L),VREAL(L),XREAL(L),YREAL(L),FEE(L),PSI(L),
1 XIS(MX),XI(L)
40 CONTINUE
IF (N1.EQ.M1) GO TO 60
READ (M1) KK,NN
J = 1
DO 50 L = 1,NN
READ (M1) UR2,VR2,XR2,YR2,PHIR2,PSIR2 ,XIS(MX+2),XIS(MX+2)
UREAL(L) = (4.*UREAL(J)-UR2)/3.
VREAL(L) = (4.*VREAL(J)-VR2)/3.

```

```

XREAL(L) = (4.*XREAL(J)-XR2)/3.
YREAL(L) = (4.*YREAL(J)-YR2)/3.
FEE(L) = (4.*FEE(J)-PHIR2)/3.
PSI(L) = (4.*PSI(J)-PSIR2)/3.
XI(L) = XI(J)
J = J+1
50 CONTINUE
60 LN = NN-2
  IF (LN.LE.1) GO TO 80
  DO 70 J = 2, LN
    CALL INTERP (J-1, MX, ICHK)
    IF (ICHK.EQ.0) GO TO 70
C   A BODY POINT HAS BEEN FOUND IN THE INTERVAL
    ILINE(MX) = MX+1
    MX = MX+1
    IF (MX.GE.MXMAX) GO TO 150
    GO TO 120
  70 CONTINUE
C   SUPERSONIC PATH WITH NO BODY POINTS
  80 IF (NROW.NE.1) GO TO 90
    WRITE (N4,200)
    MRQ = NN/65+1
  90 IF (MOD(NROW-1,MRQ).NE.0) GO TO 110
C   CONSTRUCT P-N TRIANGLE FOR SUPERSONIC PATH
    NCOUNT = 0
    DO 100 J = 1, NN, MRQ
      NCOUNT = NCOUNT+1
      L = 1HP
      IF (PSI(J).LT.0) L = 1HN
  100 ILINE(NCOUNT) = L
      WRITE (N4,180) (ILINE(J), J = 1, NCOUNT)
  110 IF ((IPL.EQ.3).OR.(IPL.EQ.4).OR.(IPL.EQ.8).OR.(IPL.EQ.9)) CALL
      1PLTML(NROW, NN)
      NROW = NROW+1
      IF (NN.GT.1) GO TO 30
C   TRIANGLE IS COMPLETE
      ILINE(MX-1) = 0
      NROW = 1
      GO TO 30
C   PLOT EVERY MRP CHARACTERISTIC FROM THE BODY TO THE SONIC LINE
  120 IF ((IPL.LE.2).OR.(IPL.EQ.5).OR.(IPL.EQ.6).OR.(IPL.EQ.7).OR.
      1(MOD(NROW-1,MRX).NE.0)) GO TO 80
      CALL CPLOT(ROTATE*CMPLX(XBODY(MX-1),YBODY(MX-1)),3)
      DO 140 L = J, LN
  140 CALL CPLOT(ROTATE*CMPLX(XREAL(L+2),YREAL(L+2)),2)
      GO TO 80
  150 WRITE (N4,190)
      CALL EXIT
  160 MODE = -1
      IF ((YR.LT.0.).AND.(NB.GT.1)) CALL CPLOT(CMPLX(0.,+YR),-3)
      CALL INIT (CHAR,XI,XI,XIS)
      MX = MX-1
      LL = NK+3
      ILINE(1) = 2
      QS = UBODY(1)*UBODY(1) + VBODY(1)*VBODY(1)
      DO 175 J = 1, NK
        ILINE(J+1) = J+2

```

```

      QSOLD = QS
      QS = UBODY(J+1)*UBODY(J+1) + VBODY(J+1)*VBODY(J+1)
      IF (QS.GT.QTOL) GO TO 165
C     SKIP THE STAGNATION POINT
      MX = MX-1
      ILINE(J) = J+2
      GO TO 175
165  IF ((QS-QCRIT)*(QSOLD-QCRIT).GT.0.) GO TO 175
C     WE HAVE JUST CROSSED THE SONIC LINE
      IF (QS.LT.QCRIT) GO TO 170
C     PIECE IN THE SUPERSONIC POINTS
      JF = J
      ILINE(J) = LL
      GO TO 175
C     REENTERING SUBSONIC REGION
170  LN = LL
      LL = ILINE(LL)
      IF (LL.NE.0) GO TO 170
      LL = LN+1
      ILINE(LN) = J+1
      MX = MX-(J-JF)
175  CONTINUE
      IF((IPL.EQ.2).OR.(IPL.EQ.4).OR.(IPL.EQ.7).OR.(IPL.EQ.9)) CALL
1     ORIGIN(3.5,2.0)
      RETURN
180  FORMAT (2X,65(1X,A1))
190  FORMAT (///5X48H***** TOO MANY POINTS DEFINING THE AIRFOIL *****
1     //5X36H **** PROGRAM STOPPED IN BODYPT **** )
200  FORMAT(1H1//51H  POSITIVE AND NEGATIVE VALUES OF THE STREAM FUNCT
1     24HION AT SUPERSONIC POINTS ///)
      END

```

```

C     SUBROUTINE PLTML(K1,K2)
      PLOTS CHARACTERISTICS IN THE SUPERSONIC REGION
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /F/ ROTATE,XAIMG,XBIMG,YAIMG,YBIMG,CL,DF
      COMMON /G/ N1,N3,N4,N7,M1
      REAL MACHN,KAPPA
      COMPLEX XI,XIBODY,CHAR,XIS
      COMMON CHAR(61,20),XI(300),XIS(300),XIBODY(300),UBODY(300),
1     VBODY(300),XBODY(300),YBODY(300),H(300),THETA(300),ANGL(300),
2     MACHN(300),KAPPA(300),XREAL(300),YREAL(300),S(300),ES(600),Q(600)
3     , PHI(600),QL(300),ILINE(300)
      COMMON UREAL(300),VREAL(300),PSI(600),PSITMP(61),MPDINT(61)
      COMPLEX ROTATE
      DATA NAR/61/ , MAR/20/
      L = 0
C     CHECK FOR FIRST ROW OF TRIANGLE
      IF (K1.GT.1) GO TO 30
      FAC = -PSI(1)/ABS(PSI(1))
      MRQ=2
10  DO 20 J = 1,K2,MRQ

```

```

      L = L+1
      CHAR(L,1) = ROTATE*CMPLX(XREAL(J),YREAL(J))
      PSITMP(L) = PSI(J)
20  MPOINT(L) = 1
      LL = L-1
C    CHECK TO SEE IF WE HAVE EXCEEDED THE DIMENSION OF CHAR
      IF (L.LE.NAR) RETURN
      WRITE (N4,100)
      MRQ = MRQ+MRQ
      L = 0
      GO TO 10
30  DO 70 J = 1,K2,MRQ
      L = L+1
      IF (FAC* PSI(J).LE.0.) GO TO 60
C    POINT IS OUTSIDE THE BODY
      MPOINT(L) = MPOINT(L)+1
      IF (MPOINT(L).EQ.2) GO TO 50
C    CHECK FOR DIMENSION OF CHAR
      IF (MPOINT(L).LE.MAR) GO TO 60
      MPOINT(L) = 2
      CALL CPLOT(CHAR(L,1),3)
      DO 40 K = 2,MAR
40  CALL CPLOT (CHAR(L,K),2)
      CHAR(L,1) = CHAR(L,MAR)
      GO TO 60
C    FIND A POINT ON THE BODY BY LINEAR INTERPOLATION
50  R2 = PSI(J)-PSITMP(L)
      R1 = PSI(J)/R2
      R2 = -PSITMP(L)/R2
      CHAR(L,1) = R1*CHAR(L,1)+ROTATE*R2*CMPLX(XREAL(J),YREAL(J))
C    STORE THE POINT FOR PLOTTING
60  K = MPOINT(L)
      CHAR(L,K) = ROTATE*CMPLX(XREAL(J),YREAL(J))
70  PSITMP(L) = PSI(J)
      IF (K2.GT.1) RETURN
C    WE HAVE FINISHED THIS SUPERSONIC PATH
C    FINISH PLOTTING THE CHARACTERISTICS
      DO 90 J = 1,LL
      L = MPOINT(J)
      IF (L.LE.1) GO TO 90
      CALL CPLOT(CHAR(J,1),3)
      DO 80 K = 2,L
80  CALL CPLOT(CHAR(J,K),2)
90  CONTINUE
      RETURN
100 FORMAT(27H TOO MANY SUPERSONIC. POINTS. ).
      END

      SUBROUTINE INTERP(K1,K2,K3)
C    COMPUTES SUPERSONIC POINTS ON BODY BY QUADRATIC INTERPOLATION
C    THE BODY IS THE STREAMLINE PSI = 0.
C    K3 IS ZERO WHEN THERE IS NO BODY POINT IN THIS INTERVAL
      REAL MACHN,KAPPA

```



```

COMPLEX XI,XIBODY,CHAR,XIS
COMMON CHAR(61,20),XI(300),XIS(300),XIBODY(300),UBODY(300),
1 VBODY(300),XBODY(300),YBODY(300),H(300),THETA(300),ANGL(300),
2 MACHN(300),KAPPA(300),XREAL(300),YREAL(300),S(300),ES(600),Q(600)
3 , PHI(600),QL(300),ILINE(300)
COMMON UREAL(300),VREAL(300),PSI(300)
COMPLEX XI13,XI23
K3 = 0
XB = PSI(K1+1)
XC = PSI(K1+2)
C CHECK FOR A SIGN CHANGE BETWEEN THE SECOND AND THIRD INTERVAL
IF (XB*XC.LT.0.) GO TO 20
IF ((XB*XC.GT.0.).AND.(K1.GT.1)) RETURN
IF ((XC.EQ.0.).AND.(XB.NE.0.)) GO TO 20
IF (XB*PSI(K1).LT.0.) GO TO 20
IF ((XB.NE.0.).OR.(XC.EQ.0.)) RETURN
IF ((K1.EQ.1).AND.(PSI(K1).EQ.0.)) RETURN
20 K3 = 1
XA = PSI(K1)
EA = XC-XB
EB = XC-XA
EC = XB-XA
C DO LINEAR INTERPOLATION BETWEEN POINTS 2 AND 3
U23 = (UREAL(K1+1)*XC-UREAL(K1+2)*XB)/EA
V23 = (VREAL(K1+1)*XC-VREAL(K1+2)*XB)/EA
X23 = (XREAL(K1+1)*XC-XREAL(K1+2)*XB)/EA
Y23 = (YREAL(K1+1)*XC-YREAL(K1+2)*XB)/EA
XI23 = (XI(K1+1)*XC-XI(K1+2)*XB)/EA
C DO LINEAR INTERPOLATION BETWEEN POINTS 1 AND 3
U13 = (UREAL(K1) *XC-UREAL(K1+2)*XA)/EB
V13 = (VREAL(K1) *XC-VREAL(K1+2)*XA)/EB
X13 = (XREAL(K1) *XC-XREAL(K1+2)*XA)/EB
Y13 = (YREAL(K1) *XC-YREAL(K1+2)*XA)/EB
XI13 = (XI(K1)*XC-XI(K1+2)*XA)/EB
C IF PSI IS NOT MONOTONIC USE LINEAR INTERPOLATION
IF (EA*EC.GT.0.) GO TO 30
K3 = -1
UBODY(K2) = U23
VBODY(K2) = V23
XBODY(K2) = X23
YBODY(K2) = Y23
XIBODY(K2) = CONJG(XI23)
RETURN
C DO QUADRATIC INTERPOLATION
30 UBODY(K2) = (U13*XB-U23*XA)/EC
VBODY(K2) = (V13*XB-V23*XA)/EC
XBODY(K2) = (X13*XB-X23*XA)/EC
YBODY(K2) = (Y13*XB-Y23*XA)/EC
XIBODY(K2) = (CONJG(XI13)*XB-CONJG(XI23)*XA)/EC
RETURN
END

```

```

SUBROUTINE BLADE (KP,NPTS,TITLE)
C  SORTS AND PLOTS AIRFOIL COORDINATES AND PRESSURE DISTRIBUTION
REAL MACH
COMMON /A/ GAMMA,MACH
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KB,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMPLEX ROTATE
COMMON /F/ ROTATE,XAIMG,XBIMG,YAIMG,YBIMG,CL,DF
COMMON /G/ N1,N3,N4,N7,M1
COMPLEX XITAIL,BP,XIA,XIB,XIC
COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
REAL MACHA,MACHB,MTAIL
COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
REAL MACHN,KAPPA
COMPLEX XI,XIBODY,CHAR,XIS
COMMON CHAR(61,20),XI(300),XIS(300),XIBODY(300),UBODY(300),
1 VBODY(300),XBODY(300),YBODY(300),H(300),THETA(300),ANGL(300),
2 MACHN(300),KAPPA(300),XREAL(300),YREAL(300),S(300),ES(600),Q(600)
3 , PHI(600),QL(300),ILINE(300)
COMMON XX(300),XP(300),XPP(300),XPPP(300),JLINK(300)
DIMENSION TITLE(12)
EMACH(QS) = SQRT(QS/(COSQ-((GAMMA-1.)/2.)*QS))
CP(QS) = HEADI*(RHO(QS)**GAMMA-PINF)
IF (KP.LE.1) RETURN
GAMX = (GAMMA-1.)/2.
COSQ = GAMX+1./(MACH*MACH)
QSB = COSQ*(MACHB*MACHB)/(1.+GAMX*(MACHB*MACHB))
PINF = RHO(QSB)**GAMMA
HEADI = 2./(GAMMA*(MACHB*MACHB)*PINF)
RAD = 180./PI
SF = 5.
CALL SETSF(SF)
IPL = MOD(IPLT/10,10)
YORIG = 2.0
IF (NB.NE.1) YORIG = 1.5
CPDR = (6.5-YORIG)/SF
EMDR = (4.5-YORIG)/SF
SM = 5./SF
IF (NB.NE.1) EMDR = (6.0-YORIG)/SF
IF (NB.NE.1) SM = 10./(3.333*SF)
IF ((IPL.LT.5).AND.(NB.EQ.1)) SM = 4./SF
IF (IPL.GT.5) SM = -2.5/SF
ANGROT = ATAN2(-XBIMG,-YBIMG)
COSA = REAL(ROTATE)
SINA = AIMAG(ROTATE)
SOL = (PI+PI)*SQRT(XBIMG*XBIMG+YBIMG*YBIMG)
DX = -(PI+PI)*((XAIMG+XBIMG)*COSA-(YAIMG+YBIMG)*SINA)
DY = -(PI+PI)*((XAIMG+XBIMG)*SINA+(YAIMG+YBIMG)*COSA)
IF (MODE.LT.0) GO TO 15
C  CREATE A POINTER LIST
DO 10 J = 1,KP
10 ILINE(J) = J+1
C  ORDER THE POINTS AROUND THE BODY AND FIND THE THICKNESS/CHORD
15 DO 20 J = 1,300
20 JLINK(J) = J
TC = 0.
LMAX = 1

```

```

J = 1
DO 30 JX = 2,KP
JM = J
K = JLINK(JM)
J = ILINE(J)
LMAXP = LMAX
XMX = XBODY(K)
LMAX = JM
L = J
DO 25 LX = JX,KP
LL = JLINK(L)
IF (XBODY(LL).LE.XMX) GO TO 25
XMX = XBODY(LL)
LMAX = L
25 L = ILINE(L)
IF (LMAX.EQ.JM) GO TO 30
JLINK(JM) = JLINK(LMAX)
JLINK(LMAX) = K
LMAX = JLINK(JM)
TC = AMAX1(TC,ABS(YBODY(LMAX)-YBODY(LMAXP)))
30 CONTINUE
CALL CHSIZE (.07)
WRITE (N4,160) TC,CL
EMA=ABS(MACHA)
EMB=ABS(MACHB)
IF (NB.EQ.1) GO TO 40
IF(EMB.GT.EMA) WRITE(N4,110) DF,SOL
IF(EMB.LT.EMA) WRITE(N4,115) SOL
THA = 90.-ANGLA - RAD*ANGROT
THB = 90.-ANGLB - RAD*ANGROT
TURN = THA-THB
WRITE(N4,120) EMB,THB,EMA,THA,TURN
GO TO 50
40 WRITE(N4,180) EMA
50 WRITE (N4,170) DX,DY
DZ = SQRT (DX*DX+DY*DY)
J = 1
DO 60 L = 1,KP
XREAL(L) = XBODY(J)*COSA-YBODY(J)*SINA
YREAL(L) = XBODY(J)*SINA+YBODY(J)*COSA
QS = UBODY(J)*UBODY(J)+VBODY(J)*VBODY(J)
QL(L) = SQRT(QS)
EMN = EMACH(QS)
MACHN(L) = EMN
ANGL(L) = ATAN2(UBODY(J)*SINA+VBODY(J)*COSA,UBODY(J)*COSA-
1 VBODY(J)*SINA)
IF ((IPL.EQ.0).OR.(IPL.EQ.5)) GO TO 60
IF (IPL.LT.5) CPX = SM*EMN+EMOR
IF (IPL.GT.5) CPX = SM*CP(QS) + CPOR
CALL CSYMBL(CMPLX(XREAL(L),CPX),3,-1)
60 J = ILINE(J)
C PLOT MACH NUMBER AT THE TAIL POINT
IF ((IPL.EQ.0). OR.(IPL.EQ.5)) GO TO 70
CALL CHRANG (45.)
QS = UBODY(1)*UBODY(1)+VBODY(1)*VBODY(1)
IF (IPL.LT.5) EMN = SM*MACHN(1)+EMOR
IF (IPL.GT.5) EMN = SM*CP(QS) + CPOR

```

```

      CALL CSYMBL(CMPLX(XREAL(1),EMN),3,-1)
      CALL CHRANG(0.)
      YR = YREAL(1)
      IF ((YR.LT.0.).AND.(NB.GT.1)) CALL CPLOT (CMPLX(0.,-YR),-3)
70  CONTINUE
      ENCODE (60,150,TITLE) EMB,CL,DZ,TC
      IF((IPL.EQ.2).OR.(IPL.EQ.4).OR.(IPL.EQ.7).OR.(IPL.EQ.9)) CALL ORIG
      LIN(3.5,4.0)
      CALL OUTPT (KP,TITLE,ANGROT,SOL)
      IF (NB.NE.1) ENCODE (60,140,TITLE) EMB,EMA,TURN,SOL
      IF (NB.EQ.1) ENCODE (60,190,TITLE) EMB,CL,DX,DY,TC
      IF ((NRN.LT.0.).AND.(MODE.LT.0)) READ (N7,130) TITLE
      IF (IPL.EQ.0) RETURN
      CALL SPLIF(KP,S,XREAL,XP,XPP,XPPP,3,0.,3,0.)
      CALL INTPL(NPTS,ES,XX,S,XREAL,XP,XPP,XPPP)
      IPEN = 3
      CALL ORIGIN (3.5,YORIG)
      DO 80 L = 1,NPTS
      QS = QR*Q(L)*Q(L)
      IF (IPL.LT.5) EMN = SM*EMACH(QS)+EMOR
      IF (IPL.GT.5) EMN = SM*CP(QS) + CPOR
      CALL CPLOT(CMPLX(XX(L),EMN),IPEN)
80  IPEN = 2
      CALL CHRANG(0.)
      CALL CHSIZE (.14)
      SM = SM*SF
      CALL SETSF (1.)
      WORD = 1HM
      IF(IPL.GT.5) WORD=2HCP
      IF((NB.NE.1).AND.(IPL.LT.5)) GO TO 500
      CALL CSYMBL((- .80,7.5),WORD,3)
      GO TO 501
500 CALL CSYMBL((- .80,8.2),WORD,3)
501 CONTINUE
      CALL CPLOT ((3.9,7.97),3)
      CALL CPLOT ((4.25,7.97),2)
      CALL CSYMBL((4.36,7.9),6HINPUT ,6)
      CALL CSYMBL ((4.0,7.5),8H+ OUTPUT ,8)
      CALL CSYMBL((-1.3,-.75),TITLE,60)
      IF ((NRN.LT.0.).AND.(MODE.LT.0)) CALL CSYMBL(-(1.2,.9),TITLE(7),60)
      IF ((NRN.LT.0.).AND.(MODE.LT.0)) READ (N7,130) TITLE
      IF (IPL.LE.5) GO TO 90
      CALL XYAXES((-1.0,4.5),3.0,3.5,-.40,90.,6H(F5.1))
      CPCRT = CP((COSQ+COSQ)/(GAMMA+1.))
      CALL CPLOT(CMPLX(-1.10,SM*CPCRT+4.5),3)
      CALL CPLOT(CMPLX(-.90,SM*CPCRT+4.5),2)
      GO TO 100
90  IF(NB.NE.1) GO TO 91
      CALL XYAXES((-1.0,2.5),0.,5.,.25,90.,6H(F5.2))
      GO TO 100
91  CALL XYAXES((-1.0,4.5),0.,4.,.3333,90.,6H(F5.2))
100 CALL FRAMER
      RETURN
110 FORMAT (34X,17HDIFFUSION FACTOR= ,F5.3//39X,10HGAP/CHORD= ,F6.3/)
115 FORMAT(39X,10HGAP/CHORD= ,F6.3/)
120 FORMAT (13X ,18HINLET MACH NUMBER=,F6.3,15X,17HINLET FLOW ANGLE=,
      1 F7.2//13X, 18H EXIT MACH NUMBER=,F6.3,15X, 17H EXIT FLOW ANGLE=,

```

```

      2 F7.2//35X,14HTURNING ANGLE=,F7.2/)
130 FORMAT (6A10)
140 FORMAT(4H M1=F4.3,5X4H M2=F4.3,5X7HDEL TH=F6.2,5X4HG/C=F4.2)
150 FORMAT (3H M=,F4.3,5X3HCL=,F5.3,5X3HDY=,F4.3,6X4HT/C=,F4.3)
160 FORMAT(1H1/36X,16HTHICKNESS/CHORD=,F6.4//32X,12HCoefficient ,
      1 8HOF LIFT= ,F6.4/ )
170 FORMAT (34X,3HDX=,F6.4,6X,3HDY=,F6.4 /)
180 FORMAT (40X,12HMACH NUMBER=, F6.3/)
190 FORMAT(2HM=,F4.3,4X3HCL=F5.3,4X3HDX=,F5.3,4X3HDY=F5.3,4X4HT/C=F4.3
      1)
      END

```

```

C      SUBROUTINE OUTPT (KP,TITLE,ANGROT,SOL)
      LISTS X,Y COORDINATES AND MACH NUMBERS
      DIMENSION TITLE(6)
      REAL MACH,MAVE
      COMMON /A/ GAMMA,MACH
      COMMON /C/ PI,GRID,TOL
      COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
      COMMON /G/ N1,N3,N4,N7,M1
      COMPLEX XITAIL,BP,XIA,XIB,XIC
      COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
      REAL MACHA,MACHB,MTAIL
      COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
      REAL MACHN,KAPPA
      COMPLEX XI,XIBODY,CHAR,XIS
      COMMON CHAR(61,20),XI(300),XIS(300),XIBODY(300),UBODY(300),
      1 VBODY(300),XBODY(300),YBODY(300),H(300),THETA(300),ANGL(300),
      2 MACHN(300),KAPPA(300),XREAL(300),YREAL(300),S(300),ES(600),Q(600)
      3 , PHI(600),QL(300),ILINE(300)
      COMMON SEPR(300)
      IPL = MOD(IPLT/10,10)
      RAD = 180./PI
      TOL = 1.E-5
      GAM = .5*(GAMMA-1.)
      EXP = GAMMA/(GAMMA-1.)
      CSINF = 1.+GAM*MACHB*MACHB
      HEADI = 2./((GAMMA*MACHB*MACHB)
      MAVE = .5*(ABS(MACHA)+ABS(MACHB))
      KQMIN = 1
      KQMAX = 1
      QMIN = MACHN(1)
      QMAX = QMIN
      H(1) = 0.
      THETA(1) = 0.
      S(1) = 0.
      KAPPA(KP) = 0.
      DO 70 L = 2,KP
      H(L) = 0.
      THETA(L) = 0.
      SEPR(L) = 0.
      IF (MACHN(L).GT.QMAX) KQMAX = L
      IF((XREAL(L)*XREAL(L)+YREAL(L)*YREAL(L)).LT.TOL) KQMIN=L

```

```

QMIN = AMIN1(MACHN(L),QMIN)
QMAX = AMAX1(MACHN(L),QMAX)
DX = XREAL(L)-XREAL(L-1)
DY = YREAL(L)-YREAL(L-1)
IF (ANGL(L)-ANGL(L-1).GT.PI/2.) ANGL(L) = ANGL(L)-PI
DTH = ANGL(L)-ANGL(L-1)
FAC = 1.
IF (DTH.NE.0.) FAC = .5*DTH/SIN(.5*DTH)
DS = FAC*SQRT(DX*DX+DY*DY)
S(L) = S(L-1)+DS
KAPPA(L-1) = -DTH/DS
70 CONTINUE
LTRU = KP+2
LTRL = -1
IF (RN.LE.0.) GO TO 105
L = KQMAX
IF (TRANU.LT.0.) GO TO 80
DO 75 L = KQMIN,KP
IF (XREAL(L).GE.TRANU) GO TO 80
75 CONTINUE
WRITE (N4,210)
GO TO 105
80 XTRANS = XREAL(L)
LTRU = L
CALL NASHMC (L,KP)
IF (TRANL.GT.0.) XTRANS = TRANL
DO 90 L = 1,KQMIN
IF (XREAL(L).LE.XTRANS) GO TO 100
90 CONTINUE
IF (RN.NE.0.) WRITE (N4,210)
100 LTRL = L
CALL NASHMC(LTRL,1)
105 WRITE (N4,690)
WRITE (N4,700)
LC = 18
IF (NB.EQ.1) LC = 11
INC = 1
REWIND N3
NRN = 1
WRITE (N3,730) TITLE,KP
SFAC = 2./S(KP)
DO 130 L = 1,KP
U = QL(L)*COS(ANGL(L))
V = QL(L)*SIN(ANGL(L))
ANG = RAD*ANGL(L)
CALL CHRANG (ANG)
IF (IPL.GT.0) CALL CSYMBL(CMPLX(XREAL(L),YREAL(L)),15,-1)
DELS = THETA(L)*H(L)
XS = XREAL(L)-DELS*SIN(ANGL(L))
YS = YREAL(L)+DELS*COS(ANGL(L))
CS = 1.+GAM*MACHN(L)*MACHN(L)
CP = HEAD1*((CSINF/CS)**EXP - 1.)
WRITE (N3,740) U,V,XS,YS,CP
IF ((L.LT.LTRL).OR.(L.GT.LTRU)) GO TO 110
MSG = 1H
IF (MACHN(L).LT.TOL) MSG = 10HSTAGNATION
IF ((L.EQ.LTRL+1).OR.(L.EQ.LTRU-1)) MSG = 10HTRANSITION

```

```

WRITE (N4,720) XREAL(L),YREAL(L),ANG,KAPPA(L),MACHN(L),MSG,XS,YS
INC = -1
GO TO 120
110 CONTINUE
WRITE (N4,710) XREAL(L),YREAL(L),ANG,KAPPA(L),MACHN(L)
1 , THETA(L),SEPR(L),XS,YS
120 LC = LC+1
S(L) = SFAC*S(L)-1.
IF (LC.LE.55) GO TO 130
LC = 4
WRITE (N4,200)
WRITE (N4,700)
130 CONTINUE
S(KP) = 1.
QS = U*U+V*V
RT = (1.+GAM*MAVE*MAVE)/(1.+GAM*MACHN(1)*MACHN(1))
FAC = .25/(1.+GAM*MACHN(1)*MACHN(1))
HBT = FAC*(H(KP)+1.) + 1.
HBB = FAC*(H(1)+.1) + 1.
CDF = 2.*RT**3*(THETA(KP)*QS**HBT+THETA(1)*QS**HBB)
IF (ABS(MACHA).LE.ABS(MACHB)) WRITE(N4,220) CDF
IF (NB.EQ.1) RETURN
COSTE = COS(ANGL(1))
COSBET = COS(ANGROT+PI*(ANGLA+ANGLB)/360.)
FAC = (COSBET/COSTE)*(1.+ .125*CDF*(HBB+HBT)**2/(SOL*COSBET))
CLOSS = (CDF+CDF)*FAC
IF (ABS(MACHB).GT.ABS(MACHA)) WRITE(N4,230) CLOSS
RETURN
200 FORMAT (1H1//)
210 FORMAT(/52H****TRANSITION NOT FOUND--BOUNDARY LAYER SKIPPED****)
220 FORMAT(1H0,10X,27HPROFILE DRAG COEFFICIENT = ,F6.4 )
230 FORMAT(1H0,10X,27H      LOSS COEFFICIENT = ,F6.4 )
690 FORMAT(13X,45HCOORDINATES FROM LOWER SURFACE TAIL TO UPPER
1,12HSURFACE TAIL /)
700 FORMAT(12X,1HX,7X,1HY,7X,3HANG,3X,1HK,7X,1HM,5X,5HTHETA,4X,3HSEP,5
1X,2HXS,6X,2HYS/)
710 FORMAT(F15.4,F8.4,F8.1,F6.2,F8.4,2F8.5,F7.4,F8.4)
720 FORMAT(F15.4,F8.4,F8.1,F6.2,F8.4,3X,A10,3X,F7.4,F8.4)
730 FORMAT (6A10,I5)
740 FORMAT (40Z0)
END

```

```

C SUBROUTINE NASHMC (K1,K2)
  COMPUTE THE BOUNDARY LAYER CORRECTION FROM POINT K1 TO K2
  COMMON /A/ GAMMA,MACH
  REAL MACHA,MACHB,MTAIL
  COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
  REAL MACHN,KAPPA
  COMPLEX XI,XIBODY,CHAR,XIS
  COMMON CHAR(61,20),XI(300),XIS(300),XIBODY(300),UBODY(300),
1 VBODY(300),XBODY(300),YBODY(300),H(300),THETA(300),ANGL(300),
2 MACHN(300),KAPPA(300),XREAL(300),YREAL(300),S(300),ES(600),Q(600)
3 , PHI(600),QL(300),ILINE(300)

```

```

COMMON SEPR(300)
REAL MH,MHSQ,NU,MACH,MAVE
DATA TR /-.3424/, RTHO/320./, TE1/5.E-3/, TE2/5.E-5/, PIMIN
1 /-1.5/, PIMAX/1.E4/
GAM = (GAMMA-1.)/2.
GAM1 = 1./(GAMMA-1.)
MAVE = .5*(ABS(MACHA)+ABS(MACHB))
CSIINF = 1.+GAM*MAVE*MAVE
INC = ISIGN(1,K2-K1)
GE = 6.5
L = K1
DSOLD = ABS(S(L)-S(L-INC))
10 LP = L+INC
MH = .5*(MACHN(L)+MACHN(LP))
MHSQ = MH*MH
CSIH = 1.+GAM*MHSQ
DS = ABS(S(LP)-S(L))
IF (LP.NE.K2) DQDS = (MACHN(LP)-MACHN(L))/(DS*MH*CSIH)
T = CSIINF/CSIH
RHOH = T**GAM1
NU = T*(1.+TR)/(RHOH*(T+TR))
RTH = RN*MH/(MAVE*NU)
IF (L.NE.K1) GO TO 30
THETAH = RTHO/RTH
THT = THETAH
THETA(L) = THETAH
30 FC = 1.0+.066*MHSQ-.008*MH*MHSQ
FR = 1.-.134*MHSQ+.027*MHSQ*MH
C DO AT MOST 200 ITERATIONS
DO 140 J = 1,200
RTAU = 1./(FC*(2.4711*ALOG(FR*RTH*THETAH)+4.75)+1.5*GE+1724./
1 (GE*GE+200.)-16.87)
TAU = RTAU*RTAU
HB = 1./(1.-GE*RTAU)
HH = (HB+1.)*(1.+.178*MHSQ)-1.
SEP = -THETAH*DQDS
PIE = HH*SEP/TAU
PIE = AMAX1(PIMIN,AMIN1(PIMAX,PIE))
G = 6.1*SQRT(PIE+1.81)-1.7
T2 = ABS(G-GE)/GE
GE = G
DT2 = DT
DT = (HH+2.-MHSQ)*SEP+TAU
IF (J.EQ.1) GO TO 110
T1 = ABS((DT-DT2)/DT)
IF ((T1.LT.TE2).AND.(T2.LT.TE1)) GO TO 130
110 THETAH = THT+.5*DT*DS
140 CONTINUE
130 THETA(LP) = DT*DS+THT
THETAH = THETA(LP)
THT = THETA(LP)
H(L) = (H(L)*DS+HH*DSOLD)/(DS+DSOLD)
H(LP) = HH
SEP = -THETAH*DQDS
SEPR(L) = (SEPR(L)*DS+SEP*DSOLD)/(DS+DSOLD)
SEPR(LP) = SEP
DSOLD = DS

```



```

L = LP
IF (L.NE.K2) GO TO 10
H(K2) = 2.*H(K2)-H(K2-INC)
H(K1) = 0.
SEPR(K2) = SEPR(K2) + (SEPR(K2)-SEPR(K2-INC))
RETURN
END

```

C

```

SUBROUTINE HOGRF (KP,TITLE)
PLOTS AIRFOIL ON LARGE SCALE
COMPLEX ROT
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMPLEX XITAIL,BP,XIA,XIB,XIC
COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
REAL MACHA,MACHB,MTAIL
COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
COMMON /L/ PLTSZ
REAL MACHN,KAPPA
COMPLEX XI,XIBODY,CHAR,XIS
COMMON CHAR(61,20),XI(300),XIS(300),XIBODY(300),UBODY(300),
1 VBODY(300),XBODY(300),YBODY(300),H(300),THETA(300),ANGL(300),
2 MACHN(300),KAPPA(300),XREAL(300),YREAL(300),S(300),ES(600),Q(600)
3 , PHI(600),QL(300),ILINE(300)
COMMON SS(900),XX(900),YY(900),YP(900),YPP(900),YPPP(900)
DATA NPTMAX /900/
IPL = MOD (IPLT,10)
IF (IPL.EQ.0) RETURN
XITAIL = XIBODY(1)
CALL LOCUS (IPL,TITLE)
IF (KP.LE.1) RETURN
NT=NK+1
DO 10 J=2,NT
10 CALL CSYMBL (XIBODY(J),3,-1)
IF (MODE.GE.0) RETURN
30 IF (PLTSZ.EQ.0.) RETURN
CALL FRAMER
CALL ORIGIN(2.75,5.0)
CALL CHSIZE (.14)
CALL CSYMBL((-0.75,-4.0),TITLE,60)
IF (RN.EQ.0) GO TO 32
RNX = 1.E-6*RN
ENCODE (20,70,ROT) RNX
CALL CSYMBL((6.5,-4.0),ROT,20)
32 CALL SETSF (ABS(PLTSZ))
CALL PGSIZE (ABS(PLTSZ)+10.,11.)
ROT = CMPLX(XREAL(KP),-YREAL(KP))
ROT = ROT/CABS(ROT)
IF(NB.EQ.1) ROT=(1.,0.)
NPTS = MIN0(NPTMAX,INT(ABS(PLTSZ)/.05 +.5))
DS = 2./FLOAT(NPTS-1)
SS(1) = -1.
DO 35 J = 2,NPTS

```

```

35 SS(J) = SS(J-1)+DS
   SS(NPTS) = 1.
   CALL SPLIF (KP,S,XREAL,YP,YPP,YPPP,3,0.,3,0.)
   CALL INTPL (NPTS,SS,XX,S,XREAL,YP,YPP,YPPP)
   CALL SPLIF (KP,S,YREAL,YP,YPP,YPPP,3,0.,3,0.)
   CALL INTPL (NPTS,SS,YY,S,YREAL,YP,YPP,YPPP)
   CALL CPLOT (ROT*CMPLX(XX(1),YY(1)),3)
   DO 40 J = 2,NPTS
40  CALL CPLOT (ROT*CMPLX(XX(J),YY(J)),2)
   IF (PLTSZ.LT.0) GO TO 55
   CALL CHSIZE (.07)
   DO 50 L = 1,KP
   CALL CSYMBL(ROT*CMPLX(XREAL(L),YREAL(L)),3,-1)
   IF (RN.LE.0.) GO TO 50
   DELS = THETA(L)*H(L)
   XS=XREAL(L)-DELS*SIN(ANGL(L))
   YS = YREAL(L) + DELS*COS(ANGL(L))
   CALL CSYMBL(ROT*CMPLX(XS,YS),4,-1)
   XREAL(L) = XS
   YREAL(L) = YS
50  CONTINUE
55  IF (RN.LE.0.) RETURN
   CALL SPLIF (KP,S,XREAL,YP,YPP,YPPP,3,0.,3,0.)
   CALL INTPL (NPTS,SS,XX,S,XREAL,YP,YPP,YPPP)
   CALL SPLIF (KP,S,YREAL,YP,YPP,YPPP,3,0.,3,0.)
   CALL INTPL (NPTS,SS,YY,S,YREAL,YP,YPP,YPPP)
   CALL CPLOT (ROT*CMPLX(XX(1),YY(1)),3)
   DO 60 J = 2,NPTS
60  CALL CPLOT (ROT*CMPLX(XX(J),YY(J)),2)
   RETURN
70  FORMAT (4HRN =,F5.1,8H MILLION)
   END

```

```

C  SUBROUTINE LOCUS(IPL,TITLE)
   PLOTS COMPLEX CHARACTERISTIC HODOGRAPH PLANE
   COMPLEX S,HSQPR,XI, CSQRT,XIMIN,XIMAX
   REAL MACH
   COMMON /A/ GAMMA,MACH
   COMMON /C/ PI,GRID,TOL
   COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
   COMPLEX XITAIL,BP,XIA,XIB,XIC
   COMMON /H/ BP(129),XIA,XIB,XIC,XITAIL,CONE,CTWO,CTHR,RATC,QR,NBPS
   REAL MACHA,MACHB,MTAIL
   COMMON /K/ MACHA,MACHB,MTAIL,ANGLA,ANGLB,ANGLT,RN,TRANU,TRANL,NRN
   DIMENSION CONL(20),TEMP(61),TITLE(12)
   COMMON DUM(1000),CONL,TEMP
   EXTERNAL CSQRT,SDFXI
   GAM = (GAMMA-1.)/2.
   COSQ = GAM+ 1./(MACH*MACH)
   IX = 5
   XCEN = 5.5
   YCEN = 6.0
   SFI = .25

```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```

XSZ = 9.
IF (NRN.GE.0) GO TO 10
IX = 10
SFI = .4
XSZ = 6.
10 IX = IX*INT(XSZ+TOL)
JX = IX
YSZ = XSZ
CALL ORIGIN (XCEN-.5*XSZ,YCEN-.5*YSZ)
XIMIN = -.5*XSZ*SFI*(1.,1.)
XIMAX = -XIMIN
IND = 1
IF (IPL.LT.5) GO TO 15
IND = 20
12 IF (COSQ.LE.8.) GO TO 15
COSQ = .25*COSQ
GO TO 12
15 DO 20 J = 1,IND
EMX = FLOAT(IND+1-J)/FLOAT(IND)
XI = COSQ*EMX*EMX/(1.+GAM*EMX*EMX)
CALL GETHSQ(XI,S,HSQPR)
20 CONL(IND+1-J) = REAL(S)
CALL CONTOR (IX,JX,IND,XSZ,YSZ,SOFXI,CONL,TEMP,XIMIN,XIMAX)
SF = 1./SFI
CALL SETSF(SF)
CALL ORIGIN (XCEN,YCEN)
CALL CSYMBL (XIA,11,-1)
CALL CSYMBL (XIB,11,-1)
CALL CSYMBL(XIC,12,-1)
CALL CSYMBL (XITAIL,26,-1)
CALL CPLOT(XIC,3)
CALL CPLOT(XIB,2)
CALL CPLOT (XIA,2)
CALL CSYMBL ((-1.,0.),63,-1)
SCF = SF
CALL SETSF (1.)
IF (NRN.GE.0) GO TO 30
CALL XYAXES ((0.,0.),3.5,3.5,SFI,0.,6H(F5.1))
CALL XYAXES ((0.,0.),3.0,3.0,SFI,90.,6H(F5.1))
CALL CSYMBL ((-3.0,-4.5),TITLE,60)
IF (MODE.LT.0) CALL CSYMBL ((-3.0,-4.9),TITLE(7),60)
GO TO 40
30 CALL XYAXES ((0.,0.),.5*XSZ,.5*XSZ,SFI,0.,6H(F5.2))
CALL XYAXES ((0.,0.),.5*YSZ,.5*YSZ,SFI,90.,6H(F5.2))
CALL CSYMBL ((-3.5,-5.2),TITLE,60)
40 CALL SETSF(SCF)
CALL CHSIZE (.07)
IF((IPL.GE.1).AND.(IPL.NE.5)) CALL PLPATH(NP,IPL+2)
RETURN
END

```

```

SUBROUTINE PLPATH(K1,K2)
C   PLOTS INTEGRATION PATHS IN COMPLEX PLANE
COMMON /C/ PI,GRID,TOL
COMMON /D/ II,IPLT,JJ,KBM,KK,LBM,MODE,MRP,NB,NC,NF,NJ,NK,NN,NP,NX
COMMON ARRAY
DATA NSYM /26/
MDE = MODE
DO 20 MODE = 1,K1
  KK = 0
  CALL GTPATH (ARRAY,NC+1,NN,K2)
  CALL LINES(NC,NN,ARRAY)
  IF (KK.LE.0) GO TO 20
  CALL GTPATH (ARRAY,NC+1,NN,K2)
  CALL LINES(NC,NN,ARRAY)
20 CONTINUE
  MODE = MDE
  IF(K2.LE.2) RETURN
  DO 30 J = 1,9
    KK = 0
    MODE = -J
    CALL GTPATH (ARRAY,NC+1,NN,K2)
    IF(MODE.LT.-9) GO TO 60
    CALL LINES(NC,NN,ARRAY)
    CALL GTPATH (ARRAY,NC+1,NN,K2)
    CALL LINES(NC,NN,ARRAY)
30 CONTINUE
60 MODE=MDE
  RETURN
END

```

```

COMPLEX FUNCTION CSQRT(Z,BRANCH)
C   COMPUTES THE COMPLEX SQUARE ROOT OF Z AND CHOOSES THE BRANCH
C   THE CUT OF THE SQUARE ROOT IS A STRAIGHT LINE FROM THE
C   ORIGIN PASSING THROUGH <-BRANCH>
DIMENSION Z(2),BRANCH(2)
REAL IMAGZ
REALZ = Z(1)
IMAGZ = Z(2)
R = REALZ*REALZ+IMAGZ*IMAGZ
Q = 0.
IF (R.EQ.0.) GO TO 40
Q = SQRT(.5*(SQRT(R)+ABS(REALZ)))
IF (REALZ.GE.0.) GO TO 10
R = Q
Q = .5*IMAGZ/R
GO TO 20
10 R = .5*IMAGZ/Q
20 IF (Q*BRANCH(1)+R*BRANCH(2)) 30,40,40
30 CSQRT = CMPLX(-Q,-R)
  RETURN
40 CSQRT = CMPLX(Q,R)
  RETURN
END

```

```

FUNCTION BUMPFN (X,FMAX,XL,XR,RAT,ALP)
C   COMPUTES A FUNCTION WHICH VANISHES FOR  $X \leq XL$  AND  $X \geq XR$ ,
C   HAS MAXIMUM VALUE FMAX AT  $X=XM$  AND HAS TWO CONTINUOUS DERIVATIVES
BUMPFN = 0.
IF ((X.LE.XL).OR.(X.GE.XR)) RETURN
HR = RAT*(XR-XL)
RR = 1.-(RAT+RAT)
DX = X-XL
T = (DX-HR)/(RR*DX+HR)
BUMPFN = FMAX*(1.-T*T)**ALP
RETURN
END

```

```

SUBROUTINE LEQ (A,B,NEQS,NSOLNS,IA,IB,DET)
C   SOLVE A SYSTEM OF LINEAR EQUATIONS OF THE FORM  $AX=B$  BY A MODIFIED
C   GAUSS ELIMINATION SCHEME
DIMENSION A(IA,IA), B(IB,IB)
NSIZ = NEQS
NBSIZ = NSOLNS
DET = 1.0
DO 40 I = 1,NSIZ
  BIG = A(I,1)
  IF (NSIZ.LE.1) GO TO 130
  DO 10 J = 2,NSIZ
    IF (ABS(BIG).GE.ABS(A(I,J))) GO TO 10
    BIG = A(I,J)
10  CONTINUE
  BG = 1.0/BIG
  DO 20 J = 1,NSIZ
20  A(I,J) = A(I,J)*BG
  DO 30 J = 1,NBSIZ
30  B(I,J) = B(I,J)*BG
  DET = DET*BIG
40  CONTINUE
  NUMSYS = NSIZ-1
  DO 120 I = 1,NUMSYS
    NN = I+1
    BIG = A(I,I)
    NBGRW = I
    DO 50 J = NN,NSIZ
      IF (ABS(BIG).GE.ABS(A(J,I))) GO TO 50
      BIG = A(J,I)
      NBGRW = J
50  CONTINUE
    BG = 1.0/BIG
    IF (NBGRW.EQ.I) GO TO 80
    DO 60 J = I,NSIZ
      TEMP = A(NBGRW,J)
      A(NBGRW,J) = A(I,J)
60  A(I,J) = TEMP
    DET = -DET
    DO 70 J = 1,NBSIZ
      TEMP = B(NBGRW,J)

```

```

      B(NBGRW,J) = B(I,J)
70  B(I,J) = TEMP
80  DO 110 K = NN,NSIZ
      PMULT = -A(K,I)*BG
      IF (PMULT.EQ.0.) GO TO 110
      DO 90 J = NN,NSIZ
90   A(K,J) = PMULT*A(I,J)+A(K,J)
      DO 100 L = 1,NBSIZ
100  B(K,L) = PMULT*B(I,L)+B(K,L)
110  CONTINUE
120  CONTINUE
130  DO 170 NCOLB = 1,NBSIZ
      DO 160 I = 1,NSIZ
      NROW = NSIZ+1-I
      TEMP = 0.0
      NXS = NSIZ-NROW
      IF (NXS.EQ.0) GO TO 150
      DO 140 K = 1,NXS
      KK = NSIZ+1-K
140  TEMP = TEMP+B(KK,NCOLB)*A(NROW,KK)
150  B(NROW,NCOLB) = (B(NROW,NCOLB)-TEMP)/A(NROW,NROW)
160  CONTINUE
170  CONTINUE
      DO 180 I = 1,NSIZ
180  DET = DET*A(I,I)
      RETURN
      END

```

```

C      SUBROUTINE FOUFC(N,G,X,A,B)
      FOURIER COEFFICIENTS BY FAST FOURIER TRANSFORM
      COMPLEX G,EIV,QP,X,GK
      DIMENSION      G(1),X(1),      A(1),B(1)
      DATA PI/3.14159265358979/
      L      = N/2
      V      = PI/L
      EIV = CMPLX(COS(V),SIN(V))
      MX = -1
      CALL FFORMS (MX,L,G,A,B,X)
      GK = 0.
      I = 1
      DO 5 J = 1,L,2
      X(J) = CMPLX(-B(I),A(I))
      X(J+1) = X(J)*EIV
5  I = I+1
      K = L
      DO 10 J = 1,L
      QP = GK-CONJG(G(J))
      GK = GK+CONJG(G(J))-QP*X(J)
      X(J) = .5*CONJG(GK)
      GK = G(K)
10  K = K-1
      X(L+1) = CMPLX(-AIMAG(X(1)),0.)
      X(1) = CMPLX(REAL(X(1)),0.)

```

RETURN
END

```

C  SUBROUTINE FFORMS(MX,NX,G,CN,SN,Y)
DO ABS(MX) COMPLEX FAST FOURIER TRANSFORMS OF LENGTH ABS(NX)
DIMENSION CN(1),SN(1),G(1),Y(1)
COMPLEX G,Y,EIX
LOGICAL ISW,NR2
DATA PI /3.14159265358979/
N = IABS(NX)
M = IABS(MX)
IF ((N.LT.2).OR.(M.LT.1)) RETURN
FAC = -ISIGN(1,NX)
NM = N*M
NH = NM/2
NQM = NH/2
NQ = 1
IF (4*(N/4).EQ.N) NQ = 4
NS = 1
NT = N
NR = 2
IF (MX.GT.0) GO TO 40
H = 0.
DH = (PI+PI)/FLOAT(N)
NDM = (N+1)/2
DO 10 J = 1,NDM
CN(J) = COS(H)
SN(J) = SIN(H)
10 H = H+DH
K = N
DO 20 J = 1,NDM
SN(J+NDM) = -SN(J)
CN(K) = CN(J+1)
20 K = K-1
40 ISW = .TRUE.
LA = NH-ISIGN(NQM,NX)
50 NSKP = MIN0(2,NR-1)
DO 55 K = NR,NT,NSKP
IF (MOD(NT,K).EQ.0) GO TO 60
55 CONTINUE
60 ND = NT/K
NS = NS*K
NR = K
NR2 = .TRUE.
IF (NR.NE.2) NR2 = .FALSE.
NDM = ND*M
NTM = NT*M
IF (NR2) GO TO 85
NSX = NS
NQ = 1
L = NDM
IF (ISW) GO TO 75
DO 65 J = 1,NDM

```

```

65 G(J) = Y(J)+Y(NDM+J)
   DO 70 KK = 3, NR
   L = L+NDM
   DO 70 J = 1, NDM
70 G(J) = G(J)+Y(L+J)
   GO TO 110
75 DO 80 J = 1, NDM
80 Y(J) = G(J)+G(NDM+J)
   DO 82 KK = 3, NR
   L = L+NDM
   DO 82 J = 1, NDM
82 Y(J) = Y(J)+G(L+J)
   GO TO 110
85 NSX = NS/NR
   IF (ISW) GO TO 95
   DO 90 J = 1, NDM
   G(J) = Y(J)+Y(NDM+J)
90 G(NH+J) = Y(J)-Y(NDM+J)
   GO TO 110
95 DO 100 J = 1, NDM
   Y(J) = G(J)+G(NDM+J)
100 Y(NH+J) = G(J)-G(NDM+J)
   IF (NS.EQ.2) GO TO 190
110 ID = ND
   IDM = NDM
   ITM = 0
   NSQ = NS/NQ+1
   DO 185 I = 2, NSX
   ITM = ITM+NTM
   IF (I.EQ.NSQ) GO TO 140
   EIX = CMPLX(CN(ID+1), FAC*SN(ID+1))
   IF (NR2) GO TO 160
   IF (ITM.GE.NM) ITM = ITM-NM
   LD = ITM+NDM
   MM = ID
   IF (ISW) GO TO 125
   DO 115 J = 1, NDM
115 G(IDM+J) = Y(ITM+J) + EIX*Y(LD+J)
   DO 120 KK = 3, NR
   MM = MM+ID
   IF (MM.GE.N) MM = MM-N
   EIX = CMPLX(CN(MM+1), FAC*SN(MM+1))
   LD = LD+NDM
   DO 120 J = 1, NDM
120 G(IDM+J) = G(IDM+J) + EIX*Y(LD+J)
   GO TO 180
125 DO 130 J = 1, NDM
130 Y(IDM+J) = G(ITM+J) + EIX*G(LD+J)
   DO 135 KK = 3, NR
   MM = MM+ID
   IF (MM.GE.N) MM = MM-N
   EIX = CMPLX(CN(MM+1), FAC*SN(MM+1))
   LD = LD+NDM
   DO 135 J = 1, NDM
135 Y(IDM+J) = Y(IDM+J) + EIX*G(LD+J)
   GO TO 180
140 LP = NH+ISIGN(NQM, NX)

```



```

LD = NH+NDM
IF (ISW) GO TO 150
DO 145 J = 1,NDM
  G(LA+J) = Y(NH+J)+CMPLX(AIMAG(Y(LD+J)), -REAL(Y(LD+J)))
145 G(LP+J) = Y(NH+J)-CMPLX(AIMAG(Y(LD+J)), -REAL(Y(LD+J)))
  GO TO 180
150 DO 155 J = 1,NDM
  Y(LA+J) = G(NH+J)+CMPLX(AIMAG(G(LD+J)), -REAL(G(LD+J)))
155 Y(LP+J) = G(NH+J)-CMPLX(AIMAG(G(LD+J)), -REAL(G(LD+J)))
  GO TO 180
160 LP = IDM+NH
  LD = ITM+NDM
  IF (ISW) GO TO 170
  DO 165 J = 1,NDM
    G(IDM+J) = Y(ITM+J) + EIX*Y(LD+J)
165 G(LP+J) = Y(ITM+J) - EIX*Y(LD+J)
    GO TO 180
170 DO 175 J = 1,NDM
  Y(IDM+J) = G(ITM+J) + EIX*G(LD+J)
175 Y(LP+J) = G(ITM+J) - EIX*G(LD+J)
180 IDM = IDM+NDM
185 ID = ID+ND
190 NT = ND
  ISW = .NOT.ISW
  IF (ND.GT.1) GO TO 50
  MX = M
  IF (NX.GT.0) GO TO 210
  IF (ISW) RETURN
  DO 200 J = 1,NM
200 G(J) = Y(J)
  RETURN
210 ENI = 1./FLOAT(N)
  IF (ISW) GO TO 230
  DO 220 J = 1,NM
220 G(J) = ENI*Y(J)
  RETURN
230 DO 240 J = 1,NM
240 G(J) = ENI*G(J)
  RETURN
END

```

```

C SUBROUTINE SPLIF(NN,S,F,FP,FPP,FPPP,KM,VM,KN,VN)
C GIVEN S AND F AT N CORRESPONDING POINTS, COMPUTE A CUBIC SPLINE
C THROUGH THESE POINTS SATISFYING AN END CONDITION IMPOSED ON
C EITHER END. FP,FPP,FPPP WILL BE THE FIRST, SECOND AND THIRD
C DERIVATIVE RESPECTIVELY AT EACH POINT ON THE SPLINE
  DIMENSION NN(2),F(2),FP(2),FPP(2),FPPP(2),S(2)
  INDX(JJ) = JJ
  N = IABS(NN(1))
  NORM = 1
  IF (NN(1).LT.0) NORM = 0
  I = INDX(1)
  J = INDX(2)

```

```

      JJ = 2
      DS = S(J)-S(I)
      D = DS
      IF (DS.EQ.0.) GO TO 200
      DF = (F(J)-F(I))/DS
      IF (IABS(KM)-2) 10,20,30
10    U = .5
      V = 3.*(DF-VM)/DS
      GO TO 50
20    U = 0.
      V = VM
      GO TO 50
30    U = -1.
      V = -DS*VM
      GO TO 50
40    I = J
      JJ = JJ+1
      J = INDX(JJ)
      DS = S(J)-S(I)
      IF (D*DS.LE.0.) GO TO 200
      DF = (F(J)-F(I))/DS
      B = 1./(DS+DS+U)
      U = B*DS
      V = B*(6.*DF-V)
50    FPP(I) = U
      FPP(I) = V
      U = (2.-U)*DS
      V = 6.*DF+DS*V
      IF (JJ.LT.N) GO TO 40
      IF (KN-2) 60,70,80
60    V = (6.*VN-V)/U
      GO TO 90
70    V = VN
      GO TO 90
80    V = (DS*VN+FPP(I))/(1.+FP(I))
90    B = V
      D = DS
100   DS = S(J)-S(I)
      U = FPP(I)-FP(I)*V
      FPPP(I) = (V-U)/DS
      FPP(I) = U
      FP(I) = (F(J)-F(I))/DS-DS*(V+U+U)/6.
      V = U
      J = I
      JJ = JJ-1
      I = INDX(JJ-1)
      IF (JJ.GT.1) GO TO 100
      N = INDX(N)
      FPPP(N) = FPPP(N-1)
      FPP(N) = B
      FP(N) = DF+D*(FPP(N-1)+B+B)/6.
      IF (KM.GT.0) RETURN
      N = IABS(NN(1))
      FPPP(J) = 0.
      V = FPP(J)
105   I = J
      JJ = JJ+1

```

```

      J = INDX(JJ)
      DS = S(J)-S(I)
      U = FPP(J)
      FPPP(J) = FPPP(I)+.5*DS*(F(I)+F(J)-DS*DS*(U+V)/12.)
      V = U
      IF (JJ.LT.N) GO TO 105
      RETURN
200 PRINT 210
      PRINT 220
      STOP
210 FORMAT (48H0**** NON-MONOTONIC ABSCISSA FOR SPLINE FIT **** )
220 FORMAT (44H0*** PROGRAM STOPPED IN SUBROUTINE SPLIF *** )
      END

```

```

SUBROUTINE PSPLIF (M,S,F,FP,FPP,FPPP,FINT)
PERIODIC SPLINE FITTING ROUTINE
DIMENSION S(1),F(1),FP(1),FPP(1),FPPP(1),FINT(1)
CON = 0.
IF (M.LT.0) CON = FINT(1)
N = IABS(M)
K = 1
I = 1
J = 2
DS = S(2)-S(1)
D = DS
IF (DS.EQ.0.) GO TO 200
DF = 6.*(F(2)-F(1))/DS
U1 = .5
V1 = DF/(DS+DS)
U2 = .5
V2 = (DF-6.)/(DS+DS)
GO TO 20
10 I = J
  J = J+K
  DS = S(J)-S(I)
  IF (D*DS.LE.0.) GO TO 200
  DF = 6.*(F(J)-F(I))/DS
  B = 1./(DS+DS+U1)
  U1 = B*DS
  V1 = B*(DF-V1)
  B = 1./(DS+DS+U2)
  U2 = B*DS
  V2 = B*(DF-V2)
20 FP(I) = U1
   FPP(I) = V1
   FPPP(I) = U2
   FINT(I) = V2
   U1 = (2.-U1)*DS
   V1 = DF+DS*V1
   U2 = (2.-U2)*DS
   V2 = DF+DS*V2
   IF (J.NE.N) GO TO 10
   V1 = -V1/U1

```

```

V2 = (6.-V2)/U2
B1 = V1
B2 = V2
40 DS = S(J)-S(I)
DF = (F(J)-F(I))/DS
U1 = FPP(I)-FP(I)*V1
FPP(I) = U1
FP(I) = DF-DS*(V1+U1+U1)/6.
V1 = U1
U2 = FINT(I)-FPPP(I)*V2
FINT(I) = U2
FPPP(I) = DF-DS*(V2+U2+U2)/6.
V2 = U2
J = I
I = I-K
IF (J.NE.1) GO TO 40
X = -(FPP(1)-B1)/((FINT(1)-B2)-(FPP(1)-B1))
FP(1) = X
U = FPP(1) + X*(FINT(1)-FPP(1))
FPP(1) = U
FPP(N) = FPP(1)
FINT(N) = FPP(N)
FP(N) = FP(1)
FPPP(N) = FP(N)
FINT(1) = CON
60 I = J
J = J+K
DS = S(J)-S(I)
FP(J) = FP(J)+X*(FPPP(J)-FP(J))
V = FPP(J)+X*(FINT(J)-FPP(J))
FPP(J) = V
FPPP(I) = (V-U)/DS
FINT(J) = FINT(I)+.5*DS*(F(I)+F(J)-DS*DS*(U+V)/12.)
U = V
IF (J.NE.N) GO TO 60
FPPP(N) = FPPP(1)
RETURN
200 PRINT 210
PRINT 220
STOP
210 FORMAT (48H0**** NON-MONOTONIC ABSCISSA FOR SPLINE FIT **** )
220 FORMAT (45H0*** PROGRAM STOPPED IN SUBROUTINE PSPLIF *** )
END

```

```

SUBROUTINE INTPL (NX,SI,FI,S,F,FP,FPP,FPPP)
C   GIVEN S,F(S) AND THE FIRST THREE DERIVATIVES AT A SET OF POINTS
C   FIND FI(SI) AT THE NX VALUES OF SI BY EVALUATING THE TAYLOR SERIES
C   OBTAINED BY USING THE FIRST THREE DERIVATIVES
DIMENSION SI(1), FI(1), S(1), F(1), FP(1), FPP(1), FPPP(1)
DATA PT/.3333333333333333/
J = 0
DO 30 I = 1,NX
VAL = 0.

```

```

      SS = SI(I)
10  J = J+1
      TT = S(J)-SS
      IF (TT) 10,30,20
20  J = MAX0(1,J-1)
      SS = SS-S(J)
      VAL = SS*(FP(J)+.5*SS*(FPP(J)+SS*PT*FPPP(J)))
30  FI(I) = F(J)+VAL
      RETURN
      END

```

```

SUBROUTINE INTPLI(MX,XI,FI,N,X,F,FP,FPP,FPPP)
DIMENSION X(1),F(1),FP(1),FPP(1),FPPP(1),XI(1),FI(1)
REAL NEW,LEFT
DATA TOL /1.E-9 /
XI(L) WILL SATISFY F(XI(L)) = FI(L) FOR L = 1 TO ABS(MX)
F,FP,FPP,FPPP ARE THE FUNCTION AND DERIVATIVES AT THE X POINTS
NX = IABS(MX)
K = 2
L1 = 1
NEW = X(1)
FN = F(1)
FVAL = FI(1)
IF (ABS(F(1)-FI(1)).GT.TOL) GO TO 5
L1 = 2
XI(1) = X(1)
IF (NX.EQ.1) RETURN
5 DO 100 L = L1,NX
  IF ((FVAL.NE.FI(L)).OR.(L.EQ.1)) GO TO 6
  NEW = X(K)
  FN = F(K)
  IF (FP(K)*FP(K-1).GT.0.) GO TO 6
  ROOT = SQRT(FPP(K-1)**2-2.*FP(K-1)*FPPP(K-1))
  DX = -2.*FP(K-1)/(FPP(K-1)+SIGN(ROOT,FPP(K-1)))
  NEW = X(K-1)+DX
  FN = F(K-1)+DX*(FP(K-1)+DX*(.5*FPP(K-1)+DX*FPPP(K-1)/6.))
6  FVAL = FI(L)
  SGN = F(K-1)-FVAL
  IF (NEW.GT.X(K-1)) SGN = FN-FVAL
  DO 10 J = K,N
    IF (FP(J)*FP(J-1).LE.0.) GO TO 7
    IF (SGN*(F(J)-FVAL).LE.0.) GO TO 20
    GO TO 10
7  ROOT = SQRT(FPP(J-1)**2-2.*FP(J-1)*FPPP(J-1))
  DX = -2.*FP(J-1)/(FPP(J-1)+SIGN(ROOT,FPP(J-1)))
  RIGHT = X(J-1)+DX
  LEFT = AMAX1(X(J-1),NEW+TOL)
  IF (LEFT.GT.RIGHT) GO TO 10
  F2 = .5*FPP(J-1)
  F3 = FPPP(J-1)/6.
  FN = F(J-1)+DX*(FP(J-1)+DX*(F2+DX*F3))
  IF (SGN*(FN-FVAL).LE.0) GO TO 65
10 CONTINUE

```

```

      IF (MX.GT.0) GO TO 11
      MX = L-1
      RETURN
11  PRINT 499,L,FI(L)
499  FORMAT ( * TROUBLE AT *,I5,3X,E16.6)
      J = K
      GO TO 100
20  OLD = AMAX1(X(J-1),NEW+TOL)
      F2 = .5*FPP(J-1)
      F3 = FPPP(J-1)/6.
      START=OLD
      DO 40 K = 1,10
      DX = OLD-X(J-1)
      FPOLD = FP(J-1)+DX*(FPP(J-1)+.5*DX*FPPP(J-1))
      IF (ABS(FPOLD).LE.TOL) GO TO 60
      FN = F(J-1)+DX*(FP(J-1)+DX*(F2+DX*F3))
      NEW = OLD-(FN-FVAL)/FPOLD
      IF (NEW.LT.START) GO TO 60
      NEW = AMIN1(NEW,X(J))
      IF (ABS(NEW-OLD).LT.TOL) GO TO 90
40  OLD = NEW
      CALL ABORT
60  RIGHT = X(J)
      LEFT = OLD
      IF (SGN*(FN-FVAL).GT.0.) GO TO 65
      RIGHT = LEFT
      LEFT = XI(L-1)
      IF (L.EQ.1) LEFT = X(1)
65  DO 70 K = 1,50
      IF ((RIGHT-LEFT).LE.TOL) GO TO 90
      NEW = .5*(LEFT+RIGHT)
      DX = NEW-X(J-1)
      FN = F(J-1)+DX*(FP(J-1)+DX*(F2+DX*F3))
      IF ((FN-FVAL)*SGN.LE.0.) GO TO 80
      LEFT= NEW
      GO TO 70
80  RIGHT = NEW
70  CONTINUE
90  XI(L) = NEW
100 K = J
      MX = NX
      RETURN
      END

```

```

SUBROUTINE GÖPLOT(NRN,LABEL,NFRAME)
DIMENSION ID(3),LABEL(2)
COMMON /KORNPP/ SF,SIZE,ANG,XMAX,YMAX,XOR,YOR
C  INITIATE PLOT
LOGICAL ISW
DATA SF/1./,SIZE/.14/,ANG/0./,XOR/0./,YOR/0./,XMAX/11./,YMAX/11./
DATA ISW /.FALSE./
IF (ISW) RETURN
NIN = 12*(NFRAME+1)

```

```

NRUN = IABS(NRN)
ENCODE (30,30,ID) LABEL,NRUN
ID(3) = ID(3).AND..NOT.7777B
ISW = .TRUE.
IF (NRN.LT.0) GO TO 25
CALL PLOTS (NIN,ID)
RETURN
25 CALL PLOTSBL(NIN,ID)
RETURN
ENTRY FRAMER
ANG = 0.
SIZE = .14
SF = 1.
XOR = 0.
YOR = 0.
XMAX = 11.
YMAX = 11.
CALL FRAME
RETURN
ENTRY ENDPLT
CALL PLOT (0.,0.,999)
RETURN
30 FORMAT (2A10,3HRUN,I5)
END

```

```

SUBROUTINE ORIGIN(X,Y)
COMMON /KORNPP/ SF,SIZE,ANG,XMAX,YMAX,XOR,YOR
XOR = X
YOR = Y
RETURN
ENTRY PGSIZE
XMAX = X
YMAX = Y
RETURN
ENTRY CHSIZE
SIZE = X
RETURN
ENTRY CHRANG
ANG = X
RETURN
ENTRY SETSF
SF = X
RETURN
ENTRY CHNMDE
MODE= IABS(MODE-10)
RETURN
END

```

```

SUBROUTINE CONTOR (M,N,NC,XS,YS,FN,CONL,TEMP,XIMIN,XIMAX)
C   PLOTS LEVEL CURVES OF ABSOLUTE VALUE OF FN
   COMPLEX FN,W,XIMIN,XIMAX
   LOGICAL ISW,JSW,KSW
   DIMENSION CONL(1),TEMP(1)
   EQUIVALENCE (A,JA),(B,JB),(C,JC),(D,JD)
   DATA ISW /.TRUE./ , TOL /.01/
   RDXI = (REAL(XIMAX)-REAL(XIMIN))/FLOAT(M)
   ADXI = (AIMAG(XIMAX)-AIMAG(XIMIN))/FLOAT(N)
   DX = XS/FLOAT(M)
   DY = YS/FLOAT(N)
   NCL = IABS(NC)
   MP = M+1
   DO 10 I = 1,MP
   W = FN(XIMIN+FLOAT(I-1)*RDXI)
10  TEMP(I) = REAL(W)*REAL(W) + AIMAG(W)*AIMAG(W)
   YP = 0.
   DO 400 J = 1,N
   XP = 0.
   W = FN(XIMIN+CMPLX(0.,FLOAT(J)*ADXI))
   TM = REAL(W)*REAL(W) + AIMAG(W)*AIMAG(W)
   YPP = YP+DY
   DO 360 I = 1,M
   XPP = XP+DX
   W = FN(XIMIN+CMPLX(FLOAT(I)*RDXI,FLOAT(J)*ADXI))
   TP = REAL(W)*REAL(W) + AIMAG(W)*AIMAG(W)
   TOP = AMAX1(TP,TM,TEMP(I),TEMP(I+1))
   BOT = AMIN1(TP,TM,TEMP(I),TEMP(I+1))
   IF (CONL(NCL).LT.BOT) GO TO 350
   DO 40 L1 = 1,NCL
   IF(CONL(L1).GT.BOT) GO TO 50
40  CONTINUE
   GO TO 350
50  DO 340 L = L1,NCL
   IF (CONL(L).GT.TOP) GO TO 350
   A = TEMP(I)-CONL(L)
   B = TEMP(I+1)-CONL(L)
   C = TM-CONL(L)
   D = TP-CONL(L)
   IA = ISIGN(1,JA)
   IB = ISIGN(2,JB)
   IC = ISIGN(4,JC)
   ID = ISIGN(8,JD)
   ITYP = 8+((IA+IB+IC+ID+1)/2)
   GO TO (340,220,230,240,250,260,270,280,280,270,260,250,240,230,
1  220,340) ITYP
220 X1 = XP
   Y1 = YP+DY*A/(A-C)
   X2 = XP+DX*A/(A-B)
   Y2 = YP
   GO TO 300
230 X1 = XP+DX*A/(A-B)
   Y1 = YP
   X2 = XPP
   Y2 = YP+DY*B/(B-D)
   GO TO 300
240 X1 = XP

```


REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```

Y1 = YP+DY*A/(A-C)
X2 = XPP
Y2 = YP+DY*B/(B-D)
GO TO 300
250 X1 = XP
Y1 = YP+DY*A/(A-C)
X2 = XP+DX*C/(C-D)
Y2 = YPP
GO TO 300
260 X1 = XP+DX*A/(A-B)
Y1 = YP
X2 = XP+DX*C/(C-D)
Y2 = YPP
GO TO 300
270 ISW = .FALSE.
E = (A+D)+(B+C)
R = ABS(E/((A+D)-(B+C)))
JSW = R.LE.TOL
IF (JSW) GO TO 240
KSW = A*E.GT.0.
IF (KSW) GO TO 230
280 X1 = XP+DX*C/(C-D)
Y1 = YPP
X2 = XPP
Y2 = YP+DY*B/(B-D)
300 CALL CPLOT (CMPLX(X1,Y1),3)
CALL CPLOT (CMPLX(X2,Y2),2)
IF (ISW) GO TO 340
ISW = .TRUE.
IF (JSW) GO TO 260
IF (KSW) GO TO 250
GO TO 220
340 CONTINUE
350 TEMP(I) = TM
XP = XPP
TM = TP
360 CONTINUE
TEMP(M+1) = TP
YP = YPP
400 CONTINUE
RETURN
END

```

```

SUBROUTINE LINES (K1,K2,ARRAY)
COMPLEX ARRAY(1)
CALL CPLOT (ARRAY(K1),3)
KP = K1+1
DO 10 J = KP,K2
10 CALL CPLOT (ARRAY(J),2)
RETURN
END

```

```

SUBROUTINE POINTS (NSYM,K1,K2,ARRAY)
COMPLEX ARRAY(1)
DO 20 J = K1,K2
20 CALL CSYMBL (ARRAY(J),NSYM,-1)
RETURN
END

```

```

C SUBROUTINE CPlot (X,IPEN)
MOVE PEN TO THE POSITION DEFINED BY X WHEN X IS ON THE PAGE
COMMON /KORNPP/ SF,SIZE,ANG,XMAX,YMAX,XOR,YOR
DIMENSION X(2)
DATA XL,YL,IPENL /0.,0.,0/
XX = XOR+SF*X(1)
YY = YOR+SF*X(2)
IF (ABS(XX+XX-XMAX).GT.XMAX) GO TO 30
IF (ABS(YY+YY-YMAX).GT.YMAX) GO TO 60
IF (IPENL.NE.0) GO TO 80
10 CALL PLOT (XX,YY,IABS(IPEN))
IPENL = 0
20 XL = XX
YL = YY
IF (IPEN.GT.0) RETURN
IF (IPENL.NE.0) GO TO 100
XOR = XX
YOR = YY
RETURN
30 IPENL = -IABS(IPENL)
IF ((XX.GT.XMAX).AND.(IPEN.LT.0.)) GO TO 90
IF (IPENL.NE.0) GO TO 20
IPENL = -2
IF (IPEN.EQ.3) GO TO 20
XP = XMAX
IF (XX.LE.0.) XP = 0.
40 YP = YY+((YL-YY)/(XL-XX))*(XP-XX)
IF (ABS(YP+YP-YMAX).GT.YMAX) GO TO 110
50 CALL PLOT (XP,YP,IABS(IPENL))
IF (IPENL.EQ.3) GO TO 10
GO TO 20
60 IPENL = IABS(IPENL)
IF (IPENL.EQ.2) GO TO 20
IPENL = 2
IF (IPEN.EQ.3) GO TO 20
YP = YMAX
IF (YY.LE.0.) YP = 0.
70 XP = XX+((XL-XX)/(YL-YY))*(YP-YY)
GO TO 50
80 IF (IABS(IPEN).EQ.3) GO TO 10
IPENL = IPENL+1
IF (IPENL.EQ.3) GO TO 70
IPENL = 3
GO TO 40
90 CALL FRAMER
XL = 0.

```

```

      YL = 0.
      IPENL = 0
      RETURN
100  XOR = XP
      YOR = YP
      RETURN
110  IF (IPENL.EQ.3) GO TO 70
      GO TO 60
      END

```

```

      SUBROUTINE CSYMBL (X,N,L)
      DIMENSION X(2)
      COMMON /KORNPP/ SF,SIZE,ANG,XMAX,YMAX,XOR,YOR
C    CHANGE RELATIVE MOVEMENTS TO ABSOLUTE INCHES
      XX = XOR+SF*X(1)
      YY = YOR+SF*X(2)
C    CHECK TO SEE IF WE ARE WITHIN THE PAGE
      IF ((XX.LT.0.).OR.(YY.LT.0.).OR.(XX.GT.XMAX).OR.(YY.GT.YMAX))
1    RETURN
      CALL SYMBOL (XX,YY,SIZE,N,ANG,L)
      RETURN
      END

```

```

C    SUBROUTINE XYAXES (X,BOT,TOP,SCF,ANGL,FORMAT)
      PLOTS AND LABELS COORDINATE AXES
      COMPLEX ZB,ZT,H,COR
      COMMON /KORNPP/ SF,SIZE,ANG,XMAX,YMAX,XOR,YOR
      DIMENSION X(2), Y(2)
      ANGO = ANG
      SFO = SF
      SIZO = SIZE
      Y(1) = XOR
      Y(2) = YOR
      ANG = 0.
      SF = 1.
      SIZE = .14
      XOR = XOR+SFO*X(1)
      YOR = YOR+SFO*X(2)
      H = 1.
      COR = (-.40,-.3)
      NC = 16
      IF (ABS(ANGL).NE.90.) GO TO 10
      H = (0.,1.)
      COR = (-.75,0.)
      NC = 15
10    CALL CPLOT (H*TOP,3)
      CALL CPLOT (-BOT*H,2)
      L = TOP
      K = BOT

```

```

      N = 1+K+L
      S = -FLOAT(K)*SCF
      ZB = -FLOAT(K)*H
      ZT = ZB+COR
      DO 20 I = 1,N
      CALL CSYMBL (ZB,NC,-1)
      B = S+FLOAT(I-1)*SCF
C      *****NON-ANSI*****
      ENCODE (10,FORMAT,A) B
      CALL CSYMBL (ZT,A,5)
      ZB = ZB+H
20  ZT = ZT+H
      SF = SFO
      SIZE = SIZE
      ANG = ANGO
      XOR = Y(1)
      YOR = Y(2)
      RETURN
      END

```

2. Update of the Analysis Code H

The basic documentation for program H is given in Volume II. The current version of the program has been improved in two ways. First, a correction term has been added to the wave drag to account for the nonconservation of mass at shock waves. Second, computing time has been dramatically reduced by adding a fast Poisson solver for the subsonic region of flow.

Two new NAMELIST input parameters, NFAST and NRELAX, have been added. The current version of program H can be run without the fast solver by setting NFAST = 0, NRELAX = 1, NS1 = 20 and running about NS = 400 crude plus NS = 200 fine grid cycles. The suggested way to run the current version is with NFAST = 1, NRELAX = 6, NS1 = 1, and NS = 20 crude plus NS = 10 fine grid cycles.

The glossary of NAMELIST input parameters on pages 183-185 of Volume II has been rewritten and is given below. It now includes the two new parameters and revised definitions of many of the others. Tables 1-3 referred to in this glossary are found on pages 188-191 of Volume II. (On page 188 the definition of EPSIL should be:

EPSIL Real. Trailing edge angle divided by pi.
 If a number greater than 1.0 is input then the
 program will compute EPSIL.)

A. Glossary of Input Parameters

ALP	Real. Angle of attack in degrees. See CL and FSYM.
BCP	Real. Starting value of the base pressure which is used when the pressure distribution is extrapolated linearly on the upper surface. BCP is not used when LSEP=M+1. Default 0.4.
BETA	Real. Damping coefficient for rotated difference scheme used to solve the flow equations. BETA greater than zero may help the convergence for Mach numbers near 1.0. Default 0.0.

/

CL	Real. Coefficient of lift. The program permits either ALP or CL to be input on a NAMELIST card. If neither ALP nor CL is input the default is $ALP = 0.0$.
EM	Real. The free stream Mach number. It must be less than 1.0. Default 0.75.
FSYM	Real. Indicates format of input airfoil coordinates to be copied onto the file TAPE3. See Table 1. The program assumes that the input airfoil coordinates are oriented at $ALP = 0.0$. Default 1.0.
GAMMA	Real. Gas constant. Default 1.4.
IS	Integer. Number of smoothings of input airfoil coordinates. Also the number of smoothings of the displacement thickness. Default 2.
ITYP	Integer. ITYP is used along with NS on NAMELIST input cards to indicate mode of operation (see Table 3). If NS is positive, ITYP=4 produces all printed and plotted output ITYP=5 same as ITYP=4 but sonic line omitted on plot ITYP=3 produces all printed output ITYP=2 produces only the Mach chart ITYP=1 produces no output ITYP=0 causes the program to terminate and produces all printed and plotted output. Default 1.
IZ	Integer. Width of output line control. IZ is the number of characters on a line of output. In addition, if $IZ = 120$, the Fourier coefficients of the mapping are printed. Default 125.
KP	Integer. Print parameter. The program prints one line of output every KP cycles. Default 1.
LL	Integer. Index of location on airfoil where the sweep through the upper and lower surfaces begins for the finite difference scheme. LL should be changed if its value lies in or near a supersonic region. Smaller values are used for high angles of attack. Default $M/2 + 1$.
LSEP	Integer. Index of x which gives the location where the optional linear extrapolation for the pressure distribution is begun on the upper surface. It should be placed at the point of separation. If used, the pressure distribution is modified from x at LSEP+1 to the trailing edge. If $LSEP = M+1$ then the pressure distribution is not altered. Default $M+1$.
M	Integer. The number of mesh intervals in the angular direction in the circle plane at which the flow equations are solved. Default 160.
N	Integer. The number of mesh intervals in the radial direction in the circle plane at which the flow equations are solved. Default 30.

NFAST	Integer. The number of sweeps through the grid points for each flow cycle using the fast Poisson solver for the subsonic region of the flow. (See NS for definition of flow cycle.) Default 1.
NFC	Integer. The number of Fourier coefficients used for the mapping. Default 80.
NPTS	Integer. The number of points at which the Nash-Macdonald boundary layer equation is solved. Default 81.
NRELAX	Integer. The number of sweeps through the grid points for each flow cycle using the relaxation technique. (See NS for definition of flow cycle.) Default 6.
NRN	Integer. Run number. Default 1.
NS	Integer. NS is used along with ITYP on NAMELIST input cards to indicate mode of operation (see Table 3). Also, if NS and ITYP are both positive, NS is the maximum number of flow cycles computed before the next NAMELIST is read. A flow cycle consists of NFAST fast solver iterations plus NRELAX relaxation iterations. Default 1.
NS1	Integer. Number of flow cycles computed between boundary layer calculations. (See NS for definition of flow cycle.) Default 1.
PCH	Real. Chord location at which the turbulent boundary layer calculation is begun (the laminar boundary layer is neglected). Transition is assumed to occur at this point. Default 0.07.
RBCP	Real. Relaxation parameter for iterating BCP. RBCP is not used when LSEP = M+1. Default 0.1.
RCL	Real. Relaxation parameter for the circulation or the angle of attack. Default 1.0.
RDEL	Real. Relaxation parameter for the boundary layer displacement thickness. Default 0.125.
RFLO	Real. Relaxation parameter for the velocity potential in the flow calculation. Default 1.4.
RN	Real. Reynolds number based on chord. If RN = 0.0 inviscid flow is computed around the input airfoil with no boundary layer. Default 20.0E6.
SEPM	Real. Bound imposed on separation parameter SEP for x less than ABS(XSEP). Also, separation is predicted when SEP is greater than SEPM. Default 0.004.
ST	Real. Convergence tolerance on the maximum velocity potential correction and the maximum circulation correction. ST = 1.E-5 may be reasonable. ST = 0.0 ensures the completion of NS flow cycles. Default 0.0.

- XMON Real. x location where the search for monotonicity of the pressure distribution is begun when modifying BCP for the pressure extrapolation. XMON is not used when $LSEP = M+1$. Default 0.95.
- XP Real. Indicator for test data. If XP is greater than zero then test data must be prepared as shown in Table 2 and copied onto the file TAPE4. The points will be plotted on the pressure distribution plot. If $XP = 0.0$, no test data are expected. Default 0.0.
- XSEP Real. Absolute value: For x less than $ABS(XSEP)$, if SEP exceeds SEPM then the program sets SEP equal to SEPM on the upper surface so that the boundary layer calculation can proceed through a shock wave. For x greater than $ABS(XSEP)$, SEP is free to exceed SEPM to allow separation to be properly predicted. Sign: Make XSEP positive for supercritical airfoils. On the upper surface the displacement thickness is required to be monotonically increasing. On the lower surface for x less than 0.6 the displacement thickness is required to be monotonically increasing and after that once it starts to decrease it is required to be monotonically decreasing. Make XSEP negative for other airfoils. Then the upper and lower surfaces are both treated as upper surfaces. Default 0.93.

B. LISTING OF UPDATE CORRECTIONS FOR VOLUME II

PAGE 202 INSERT AFTER LINE 3 THE FOLLOWING
COMMON /FL/FLUXT4

PAGE 202 DELETE LINE 11 AND REPLACE BY THE FOLLOWING
4,NPTS,LL,I,LSEP,M4,NEW

PAGE 202 DELETE LINE 19 AND REPLACE BY THE FOLLOWING
2 XMON,XP,XSEP,NRELAX,NFAST

PAGE 202 INSERT AFTER LINE 22 THE FOLLOWING
NEW=1
NFAST=1
NRELAX=6
NS1=1

PAGE 202 INSERT AFTER LINE 38 THE FOLLOWING
NEW=1

PAGE 204 DELETE LINE 7 AND REPLACE BY THE FOLLOWING
105 CONTINUE
IF(NFAST.LE.0) GO TO 141
CALL SWEEP1
141 IF(NRELAX.LE.0) GO TO 151
DO 142 LF=1,NRELAX
CALL SWEEP
142 CONTINUE
151 NEW=0

PAGE 204 DELETE LINE 17 AND REPLACE BY THE FOLLOWING
1 BCP,FLUXT4

PAGE 204 DELETE LINE 55 AND REPLACE BY THE FOLLOWING
190 FORMAT(5X,I4,4E12.3,I4,I3,I6,2F10.4,2F11.5,E12.4)

PAGE 205 DELETE LINE 7 AND REPLACE BY THE FOLLOWING
1 2X,2HJK,2X3HNSP,5XA4,5X4HANGD,8X3HCPI,8X3HBCP,8X4HFLX4/)

PAGE 206 DELETE LINE 15 AND REPLACE BY THE FOLLOWING
IF((J.LE.MA).OR.(J.GE.MB)) J=J+1

PAGE 207 INSERT AFTER LINE 15 THE FOLLOWING
COMMON /FL/FLUXT4

PAGE 207 DELETE LINE 34 AND REPLACE BY THE FOLLOWING
LLP=LL+1
DO 30 I=LLP,M

PAGE 207 DELETE LINES 38 THRU 43 AND REPLACE BY THE FOLLOWING
DO 32 J=1,N

```

32 PHI(M,J)=PHI(M,J)-E(J)
   DO 51 J=1,N
     E(J)=0.
     RPP(J)=0.
51 CONTINUE

```

PAGE 208 INSERT AFTER LINE 2 THE FOLLOWING

```

   DO 11 J=1,NN
     PHI(MM+1,J)=PHI(2,J)+DPHI
     E(J)=0.
11 CONTINUE
   TE=-2.
   I=MM
   CALL MURMAN
   DO 50 J=1,N
     PHI(MM,J)=PHI(MM,J)-E(J)
50 PHI(1,J)=PHI(MM,J)-DPHI
   DO 12 J=1,N
     E(J)=0.
12 CONTINUE
   TE=2.
   I=LL
   CALL MURMAN
   DO 13 J=1,N
13 PHI(LL,J)=PHI(LL,J)-E(J)

```

PAGE 208 INSERT AFTER LINE 13 THE FOLLOWING

```

   FLUXT=0.
   NF=N-10
   IF(N.LT.30) NF=N-5
   DO 242 L=2,MM
     U=R(NF)*(PHI(L+1,NF)-PHI(L-1,NF))*DELTH-SI(L)
     V=R(NF)*R(NF)*(PHI(L,NF+1)-PHI(L,NF-1))*DELR -CO(L)
     QF=(U*U+V*V)/FP(L,NF)
     RH=(1.+C2*EM*EM*(1.-QF))*(.5/C2)
     FLUX=RH*V/R(NF)
     FLUXT=FLUXT+FLUX
242 CONTINUE
   FLUXT=DT*FLUXT*CHD
   FLUXT4=FLUXT

```

PAGE 213 INSERT AFTER LINE 16 THE FOLLOWING
COMMON /FL/FLUXT4

PAGE 214 INSERT AFTER LINE 12 THE FOLLOWING

```

   QCR=SQRT(QCRIT)
   DCD4=2.*(QCR-1.)*FLUXT4
   CD=CD+DCD4
   CDW=CDW+DCD4
   PRINT 261, CDW,CDF,CD
261 FORMAT(5H CDW=F10.5,5H CDF=F10.5,4H CD=F10.5)

```

PAGE 222 INSERT AFTER LINE 42 THE FOLLOWING

```

   SN1=2./ARCL(MM)
   DO 322 I=1,M
322 ARCL(I)=ACOS(1.-SN1*ARCL(I))
   ARCL(MM)=PI

```

PAGE 222 INSERT AFTER LINE 54 THE FOLLOWING
 $TT(1) = .5 * (TH(1) + TH(N)) + PI$

PAGE 223 DELETE LINE 19 AND REPLACE BY THE FOLLOWING

```

Z(1)=0.
VAL=.5*PILC
VAL1=PILC/3.
Z(2)=VAL*(DS(1)+DS(2))
NI=NFC+1
DO 295 J=3,NI,2
Z(J)=Z(J-2)+VAL1*(DS(J-2)+4.*DS(J-1)+DS(J))
IF(J.EQ.NI) GO TO 296
295 Z(J+1)=Z(J)+VAL*(DS(J)+DS(J+1))
296 CONTINUE
Z(MC)=0.
Z(MC-1)=VAL*(DS(MC)+DS(MC-1))
NII=NFC-2
DO 299 J=2,NII,2
MCJ=MC-J
Z(MCJ)=Z(MCJ+2)+VAL1*(DS(MCJ+2)+4.*DS(MCJ+1)+DS(MCJ))
299 Z(MCJ-1)=Z(MCJ)+VAL*(DS(MCJ)+DS(MCJ-1))
300 CONTINUE
Z1=Z(MC-NII)+VAL1*(DS(MC-NII)+4.*DS(MC-NII-1)+DS(MC-NII-2))
Z1=Z(NI+1)-Z1
DO 301 J=3,NI,2
DS1=Z(NFC+J)-Z(NFC+J-1)
Z(NFC+J-1)=Z(NFC+J-2)-Z1
IF(J.EQ.NI) GO TO 303
Z1=Z(NFC+J+1)-Z(NFC+J)
303 CONTINUE
Z(NFC+J)=Z(NFC+J-1)-DS1
301 CONTINUE

```

PAGE 230 INSERT AFTER LINE 21 THE FOLLOWING

```

C SET THE SINES AND COSINES
PI=3.14159265358979
DT = (PI+PI)/FLOAT(N)
IF((SN(1).EQ.0.).AND.(SN(2).EQ.SIN(DT))) GO TO 11
ANG = 0.
DO 5 J = 1,N
CN(J) = COS(ANG)
SN(J)=-SIN(ANG)
5 ANG = ANG+DT

```

PAGE 230 DELETE LINES 32 THRU 35 AND REPLACE BY THE FOLLOWING

```

L = IQ+J
LP=L+ND
M=ID
W = F(L)+F(LP)*CMPLX(CN(M+1),SN(M+1))
IF(NR.EQ.2) GO TO 24
L=LP
DO 26 K=3,NR

```

PAGE 230 DELETE LINE 37 AND RELACE BY THE FOLLOWING

```

M = M+ID
IF (M.GE.N) M = M-N

```

PAGE 230 DELETE LINE 41 AND REPLACE BY THE FOLLOWING

```

      IQ=IQ+NQ
      IF(IQ.GE.N) IQ=IQ-N
22 CONTINUE

```

PAGE 231 DELETE LINES 6 THRU 9 AND REPLACE BY THE FOLLOWING

```

      L = IQ+J
      LP=L+ND
      M=ID
      W=X(L)+X(LP)*CMPLX(CN(M+1),SN(M+1))
      IF(NR.EQ.2) GO TO 74
      L=LP
      DO 76 K=3,NR

```

PAGE 231 DELETE LINE 11 AND REPLACE BY THE FOLLOWING

```

      M = M+ID
      IF (M.GE.N) M = M-N

```

PAGE 231 DELETE LINE 15 AND REPLACE BY THE FOLLOWING

```

      IQ=IQ+NQ
      IF(IQ.GE.N) IQ=IQ-N
72 CONTINUE

```

PAGE 233 INSERT AFTER LINE 52 THE FOLLOWING

```

      IF(XSEP.GE.0.) GO TO 141
      FAC=(S(NPTS-3)-S(NPTS))/(S(NPTS-3)-S(NPTS-1))
      THETA(NPTS)=FAC*THETA(NPTS-1)+(1.-FAC)*THETA(NPTS-3)
      H(NPTS)=FAC*H(NPTS-1)+(1.-FAC)*H(NPTS-3)
      DELS(NPTS)=H(NPTS)*THETA(NPTS)
141 CONTINUE

```

PAGE 234 DELETE LINE 17 AND REPLACE BY THE FOLLOWING

```

      IF (J.GT.2) GO TO 190

```

PAGE 234 INSERT AFTER LINE 36 THE FOLLOWING

```

      IF(XSEP.GE.0.) GO TO 221
      FAC=(S(2)-S(1))/(S(2)-S(3))
      DELS(1)=FAC*DELS(3)+(1.-FAC)*DELS(2)
221 CONTINUE

```

PAGE 239 INSERT AFTER LINE 40 THE FOLLOWING FIVE SUBROUTINES

```

SUBROUTINE SWEEP1
COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CD(162)
2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TITLE(4),M,N,MM,NN,NSP
3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
4 ,NPTS,LL,I,LSEP,M4,NEW
COMMON /SOL1/ Q(162,31)

```

```

DATA Q/5022*0.0/
YR=0.
NSP=0
DO 10 J=1,NN
PHI(MM,J)=PHI(1,J)+DPHI
PHI(MM+1,J)=PHI(2,J)+DPHI
10 CONTINUE
TE=-2
DO 30 I=LL,MM
CALL MURMAN1
DO 100 J=1,N
Q(I,J)=D(J)
100 CONTINUE
30 CONTINUE
TE=2
I=LL
80 I=I-1
CALL MURMAN1
DO 60 J=1,N
Q(I,J)=D(J)
60 CONTINUE
IF(I.GT.2) GO TO 80
DO 61 J=1,N
61 Q(1,J)=Q(MM,J)
210 FORMAT(5(2I4,E16.8))
CALL SOLVI
200 FORMAT(5(I4,E16.8))
DO 110 I=1,M
DO 110 J=1,N
110 PHI(I,J)=PHI(I,J)+Q(I,J)
DO 111 J=1,N
111 PHI(MM,J)=PHI(1,J)+DPHI
IF(RCL.EQ.0.) GO TO 90
YA=RCL*((PHI(M,1)-(PHI(2,1)+DPHI))*DELTH+SI(1))
IF(MODE.EQ.1) GO TO 90
ALP=ALP-.5*YA
CALL COSI
GO TO 95
90 YA=IP*YA/(1.+BET)
DPHI=DPHI+YA
95 DO 97 L=1,M
97 PHI(L,NN)=DPHI*PHIR(L)
IF(MODE.EQ.0) RETURN
DO 120 J=1,N
DO 120 L=1,M
120 PHI(L,J)=PHI(L,J)+YA*PHIR(L)
RETURN
END

```

```

SUBROUTINE MURMAN1
COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CC(162)
2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)

```

```

3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TITLE(4),M,N,MM,NN,NSP
3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
4, NPTS,LL,I,LSEP,M4,NEW
PHIO=PHI(I,2)-2.*DR*CO(I)
PHIYP=PHI(I,2)-PHI(I,1)
PHIYY=PHIYP+PHIO-PHI(I,1)
PHIXX=PHI(I+1,1)+PHI(I-1,1)-PHI(I,1)-PHI(I,1)
PHIXM=PHI(I+1,1)-PHI(I-1,1)
PHIXP=PHI(I+1,2)-PHI(I-1,2)
IF(I.NE.MM) GO TO 10
D(1)=C1*(PHIXX+RS(1)*PHIYY+RA4*CO(I))
D(1)=-D(1)/C1
GO TO 40
10 U=PHIXM*DELTH-SI(I)
BQ=U/FP(I,1)
QS=U*BQ
J=1
IF(QS.LE.QCRIT) GO TO 30
D(1)=0.
GO TO 40
30 CONTINUE
CS=C1-C2*QS
BQ=BQ*QS*(FP(I-1,1)-FP(I+1,1))
X=RA4*(CS+QS)*CO(I)
CMQS=CS-QS
D(1)=CS*RS(1)*PHIYY+RI(1)*BQ+X+CMQS*PHIXX
D(1)=-D(1)/CS
40 CONTINUE
DO 60 J=2,N
PHIXX=PHI(I+1,J)+PHI(I-1,J)-PHI(I,J)-PHI(I,J)
DU=PHIXP
PHIXP=PHI(I+1,J+1)-PHI(I-1,J+1)
PHIXY=PHIXP-PHIXM
PHIXM=DU
DU=DU*DELTH
PHIYM=PHIYP
PHIYP=PHI(I,J+1)-PHI(I,J)
PHIYY=PHIYP-PHIYM
U=R(J)*DU-SI(I)
DV=R(J)*(PHI(I,J+1)-PHI(I,J-1))*DELR
V=DV*R(J)-CO(I)
RAV=R(J)*RA*V
BQ=1./FP(I,J)
BQU=BQ*U
US=BQU*U
UV=(BQU+BQU)*V
VS=BQ*V*V
QS=US+VS
IF(QS.LE.QCRIT) GO TO 50
D(J)=0.
GO TO 60
50 CS=C1-C2*QS
CMVS=CS-VS
CMUS=CS-US

```

```

UV1=.5*BQU*RAV
C(J)=RS(J)*CMVS
D(J)=RA4*((CMVS+US-VS)*DV-UV*DU)+RI(J)*QS*BQ*(U*(FP(I-1,J)-FP(I+1,
1J))+RAV*(FP(I,J-1)-FP(I,J+1)))+CMUS*PHIXX-UV1*PHIYY+C(J)*PHIYY
D(J)=-D(J)/CS
60 CONTINUE
RETURN
END

```

```

SUBROUTINE SOLV1
COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CC(162)
2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,OCN,DSN,RA4,EPSIL,QCRIT,C1,C2
2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
4 ,NPTS,LL,I,LSEP,M4,NEW
COMPLEX FF,F1,GG
DIMENSION CX(162),SX(162),FF(162),GG(162),F1(31)
COMMON /SOL1/ Q(162,31)
IF(NEW.NE.1) GO TO 30
DO 1 I=1,M
CX(I)=COS((I-1)*DT)
SX(I)=SIN((I-1)*DT)
1 CONTINUE
NEW=0.
MMP=MM+1
30 CONTINUE
MA=M/2
MA1=MA+1
DO 2 J=1,N,2
CALL TWOFFT(M,Q(1,J),Q(1,J+1),FF,GG,CX,SX,1)
DO 7 I=1,MA1
IM=M-I+3
Q(I,J)=REAL(FF(I))
Q(I,J+1)=REAL(GG(I))
Q(IM,J)=-AIMAG(FF(I))
Q(IM,J+1)=-AIMAG(GG(I))
7 CONTINUE
2 CONTINUE
HR=.5*DR
DO 3 J=1,N
D(J)=2.*RS(J)
T=RA*RA*R(J)
B(J)=T*(R(J)-HR)
3 C(J)=T*(R(J)+HR)
C(1)=D(1)
DO 4 I=1,MA1
IM=M-I+3
DO 5 J=1,N
A(J)=-D(J)-2.*(1.-CX(I))

```

```

5  FF(J)=CMPLX(Q(I,J),Q(IM,J))
   CALL TRID1(B,A,C,FF,F1,N,IDIM)
   DO 8 J=1,N
     Q(I,J)=REAL(F1(J))
     Q(IM,J)=AIMAG(F1(J))
8  CONTINUE
4  CONTINUE
   DO 9 J=1,N,2
     DO 10 I=1,MA1
       IM=M-I+3
       FF(I)=CMPLX(Q(I,J),-Q(IM,J))
       GG(I)=CMPLX(Q(I,J+1),-Q(IM,J+1))
       FF(IM)=CMPLX(Q(I,J),Q(IM,J))
10  GG(IM)=CMPLX(Q(I,J+1),Q(IM,J+1))
       CALL TWOFFT(-M,Q(1,J),Q(1,J+1),FF,GG,CX,SX,1)
9  CONTINUE
   DO 12 J=1,N
     Q(MM,J)=Q(1,J)
12  Q(MMP,J)=Q(2,J)
   RETURN
   END

```

```

SUBROUTINE TRID1(A,B,C,RHS,OUT,N,IDIM)
COMPLEX RHS,OUT
DIMENSION A(1),B(1),C(1)
DIMENSION GA(35),RHS(IDIM),OUT(35)
REC=1./B(1)
GA(1)=REC*C(1)
OUT(1)=REC*RHS(1)
DO 10 J=2,N
  REC=1./(B(J)-A(J)*GA(J-1))
  GA(J)=REC*C(J)
10  OUT(J)=REC*(RHS(J)-A(J)*OUT(J-1))
DO 20 JJ=2,N
  J=N-JJ+1
20  OUT(J)=OUT(J)-GA(J)*OUT(J+1)
RETURN
END

```

```

--  - SUBROUTINE TWOFFT(NS,F,G,ALP,BET,CN,SN,IDIM)
C  ABS(NS) IS THE NUMBER OF POINTS IN EACH ARRAY
C  DO FFT FOR F AND G OR REVERSE TRANSFORM FOR ALP AND BET
C  IF NS<0 THE REVERSE TRANSFORM IS PERFORMED
C  FUNCTIONS F AND G ARE REPRESENTED BY ARRAYS OF THEIR VALUES
C  ALP AND BET ARE COMPLEX FOURIER COEFFICIENTS FOR F AND G
C  ALP(N) IS OF THE FORM A(N)-I*B(N)
C  CN AND SN ARE THE COSINE AND SINE ARRAYS
C  IDIM IS THE SKIP FACTOR BETWEEN POINTS IN F AND G
C  COMPLEX ALP,BET,X

```



```

        DIMENSION F(IDIM,1),G(IDIM,1),ALP(1),BET(1),CN(1),SN(1)
        N = IABS(NS)
        L = N/2
C      SET UP AND DO COMPLEX TRANSFORM
        IF (NS.LT.0) GO TO 20
        DO 10 J = 1,N
10     ALP(J) = CMPLX(F(1,J),G(1,J))
        GO TO 40
C      SET UP FOR REVERSE TRANSFORM
20     J=N+1
        DO 30 K = 1,L
            X = -CMPLX(AIMAG(BET(K))-REAL(ALP(K)),AIMAG(ALP(K))+REAL(BET(K)))
            ALP(J)=X
            X = CMPLX(REAL(ALP(K))+AIMAG(BET(K)),AIMAG(ALP(K))-REAL(BET(K)))
            ALP(K)=X
30     J = J-1
        K=L+1
        ALP(K) = 1.*(CMPLX(REAL(ALP(K))+AIMAG(BET(K)),AIMAG(ALP(K))-REAL(BE
1T(K))))
40     CALL FFORM(N,ALP,BET,CN,SN)
C      NOW SEPARATE OUT THE REAL AND IMAGINARY PARTS
        J = N
        IF (NS.LT.0) GO TO 60
        ENI=.5
        DO 50 K = 1,L
            X = CONJG(ALP(J))-ALP(K+1)
            BET(K+1) = -ENI*CMPLX( AIMAG(X),REAL(X))
            ALP(K+1) = ENI*(CONJG(ALP(K+1))+ALP(J))
50     J = J-1
        BET(1) = (ENI+ENI)*AIMAG(ALP(1))
        ALP(1) = (ENI+ENI)*REAL(ALP(1))
        RETURN
60     DO 70 J = 1,N
            F(1,J) = REAL(ALP(J))/N
70     G(1,J) = -AIMAG(ALP(J))/N
        RETURN
        END

```

- Vol 59. J. A. Hanson, Growth in Open Economies V, 128 pages 1971
- Vol 60. H. Hauptmann, Schatz- und Kontrolltheorie in stetigen dynamischen Wirtschaftsmodellen V, 104 Seiten 1971.
- Vol 61. K. H. F. Meyer, Wartesysteme mit variabler Bearbeitungsrate VII, 314 Seiten 1971
- Vol 62. W. Krelle u. G. Gabisch unter Mitarbeit von J. Burgermeister, Wachstumstheorie VII, 223 Seiten 1972
- Vol 63. J. Kohlas, Monte Carlo Simulation im Operations Research VI, 162 Seiten 1972
- Vol 64. P. Gessner u. K. Spremann, Optimierung in Funktionenräumen IV, 120 Seiten 1972
- Vol 65. W. Everling, Exercises in Computer Systems Analysis VIII, 184 pages 1972
- Vol 66. F. Bauer, P. Garabedian and D. Korn, Supercritical Wing Sections V, 211 pages 1972
- Vol 67. I. V. Girsanov, Lectures on Mathematical Theory of Extremum Problems V, 136 pages 1972
- Vol 68. J. Loeckx, Computability and Decidability. An Introduction for Students of Computer Science VI, 76 pages 1972
- Vol 69. S. Ashour, Sequencing Theory. V, 133 pages 1972
- Vol 70. J. P. Brown, The Economic Effects of Floods. Investigations of a Stochastic Model of Rational Investment Behavior in the Face of Floods V, 87 pages 1972.
- Vol 71. R. Henn und O. Opitz, Konsum- und Produktionstheorie II V, 134 Seiten 1972
- Vol 72. T. P. Bagchi and J. G. C. Templeton, Numerical Methods in Markov Chains and Bulk Queues XI, 89 pages 1972
- Vol 73. H. Kiendl, Suboptimale Regler mit abschnittsweise linearer Struktur VI, 146 Seiten 1972.
- Vol 74. F. Pokropp, Aggregation von Produktionsfunktionen VI, 107 Seiten. 1972
- Vol 75. GI-Gesellschaft für Informatik e.V. Bericht Nr. 3. 1. Fachtagung über Programmiersprachen. München, 9.-11. März 1971. Herausgegeben im Auftrag der Gesellschaft für Informatik von H. Langmaack und M. Paul VII, 280 Seiten 1972
- Vol 76. G. Fandel, Optimale Entscheidung bei mehrfacher Zielsetzung II, 121 Seiten 1972.
- Vol 77. A. Auslender, Problemes de Minimax via l'Analyse Convexe et les Inégalités Variationnelles. Theorie et Algorithmes VII, 132 pages 1972
- Vol 78. GI-Gesellschaft für Informatik e.V. 2. Jahrestagung, Karlsruhe, 2.-4. Oktober 1972. Herausgegeben im Auftrag der Gesellschaft für Informatik von P. Deussen XI, 576 Seiten. 1973.
- Vol 79. A. Berman, Cones, Matrices and Mathematical Programming V, 96 pages 1973
- Vol 80. International Seminar on Trends in Mathematical Modelling, Venice, 13-18 December 1971. Edited by N. Hawkes VI, 288 pages 1973
- Vol 81. Advanced Course on Software Engineering. Edited by F. L. Bauer XII, 545 pages 1973
- Vol 82. R. Saeks, Resolution Space, Operators and Systems. X, 267 pages 1973
- Vol 83. NTG/GI-Gesellschaft für Informatik, Nachrichtentechnische Gesellschaft. Fachtagung „Cognitive Verfahren und Systeme“, Hamburg, 11.-13. April 1973. Herausgegeben im Auftrag der NTG/GI von Th. Einsele, W. Giloi und H.-H. Nagel VIII, 373 Seiten 1973
- Vol 84. A. V. Balakrishnan, Stochastic Differential Systems I. Filtering and Control. A Function Space Approach V, 252 pages 1973
- Vol 85. T. Page, Economics of Involuntary Transfers. A Unified Approach to Pollution and Congestion Externalities XI, 159 pages 1973
- Vol 86. Symposium on the Theory of Scheduling and its Applications. Edited by S. E. Elmaghraby. VIII, 437 pages 1973
- Vol 87. G. F. Newell, Approximate Stochastic Behavior of n-Server Service Systems with Large n VII, 118 pages 1973
- Vol 88. H. Steckhan, Güterströme in Netzen VII, 134 Seiten 1973
- Vol 89. J. P. Wallace and A. Sherret, Estimation of Product Attributes and Their Importances V, 94 pages 1973
- Vol 90. J.-F. Richard, Posterior and Predictive Densities for Simultaneous Equation Models VI, 226 pages 1973
- Vol 91. Th. Marschak and R. Selten, General Equilibrium with Price-Making Firms XI, 246 pages 1974
- Vol 92. E. Dierker, Topological Methods in Walrasian Economics IV, 130 pages. 1974
- Vol 93. 4th IFAC/IFIP International Conference on Digital Computer Applications to Process Control, Part I. Zurich/Switzerland, March 19-22, 1974. Edited by M. Mansour and W. Schauffelberger XVIII, 544 pages 1974.
- Vol 94. 4th IFAC/IFIP International Conference on Digital Computer Applications to Process Control, Part II. Zurich/Switzerland, March 19-22, 1974. Edited by M. Mansour and W. Schauffelberger XVIII, 546 pages 1974
- Vol 95. M. Zeleny, Linear Multiojective Programming X, 220 pages 1974
- Vol 96. O. Moeschlin, Zur Theorie von Neumannscher Wachstumsmodelle. XI, 115 Seiten. 1974
- Vol 97. G. Schmidt, Über die Stabilität des einfachen Bedienungskanals VII, 147 Seiten 1974
- Vol 98. Mathematical Methods in Queueing Theory. Proceedings 1973. Edited by A. B. Clarke VII, 374 pages 1974.
- Vol 99. Production Theory. Edited by W. Eichhorn, R. Henn, O. Opitz, and R. W. Shephard VIII, 386 pages 1974
- Vol 100. B. S. Duran and P. L. Odell, Cluster Analysis. A Survey VI, 137 pages 1974
- Vol 101. W. M. Wonham, Linear Multivariable Control. A Geometric Approach X, 344 pages 1974.
- Vol 102. Analyse Convexe et Ses Applications. Comptes Rendus, Janvier 1974. Edited by J.-P. Aubin IV, 244 pages 1974
- Vol 103. D. E. Boyce, A. Farhi, R. Weischedel, Optimal Subset Selection. Multiple Regression, Interdependence and Optimal Network Algorithms XIII, 187 pages 1974
- Vol 104. S. Fujino, A Neo-Keynesian Theory of Inflation and Economic Growth V, 96 pages 1974
- Vol 105. Optimal Control Theory and its Applications. Part I. Proceedings 1973. Edited by B. J. Kirby VI, 425 pages 1974
- Vol 106. Optimal Control Theory and its Applications. Part II. Proceedings 1973. Edited by B. J. Kirby VI, 403 pages 1974
- Vol 107. Control Theory, Numerical Methods and Computer Systems Modeling. International Symposium, Rocquencourt, June 17-21, 1974. Edited by A. Bensoussan and J. L. Lions VIII, 757 pages 1975
- Vol 108. F. Bauer et al., Supercritical Wing Sections II. A Handbook V, 296 pages 1975
- Vol 109. R. von Randow, Introduction to the Theory of Matroids IX, 102 pages 1975
- Vol 110. C. Striebel, Optimal Control of Discrete Time Stochastic Systems III, 208 pages 1975
- Vol 111. Variable Structure Systems with Application to Economics and Biology. Proceedings 1974. Edited by A. Ruberti and R. R. Mohler VI, 321 pages 1975.
- Vol 112. J. Wihlem, Objectives and Multi-Objective Decision Making Under Uncertainty IV, 111 pages 1975.
- Vol 113. G. A. Aschinger, Stabilitätsaussagen über Klassen von Matrizen mit verschwindenden Zeilensummen V, 102 Seiten 1975.
- Vol 114. G. Uebe, Produktionstheorie XVII, 301 Seiten 1976

- Vol 115 Anderson et al., Foundations of System Theory Finitary and Infinitary Conditions VII, 93 pages 1976
- Vol 116 K. Miyazawa, Input-Output Analysis and the Structure of Income Distribution. IX, 135 pages 1976
- Vol 117: Optimization and Operations Research Proceedings 1975 Edited by W. Oettli and K. Ritter IV, 316 pages 1976
- Vol 118 Traffic Equilibrium Methods, Proceedings 1974. Edited by M. A. Florian XXIII, 432 pages. 1976.
- Vol. 119 Inflation in Small Countries Proceedings 1974. Edited by H. Frisch VI, 356 pages. 1976.
- Vol 120: G. Hasenkamp, Specification and Estimation of Multiple-Output Production Functions VII, 151 pages 1976.
- Vol. 121: J. W. Cohen, On Regenerative Processes in Queueing Theory IX, 93 pages. 1976
- Vol. 122: M. S. Bazaraa, and C. M. Shetty, Foundations of Optimization VI 193 pages 1976
- Vol. 123. Multiple Criteria Decision Making Kyoto 1975 Edited by M. Zeleny. XXVII, 345 pages 1976
- Vol. 124: M. J. Todd. The Computation of Fixed Points and Applications. VII, 129 pages 1976
- Vol 125 Karl C Mosler. Optimale Transportnetze Zur Bestimmung ihres kostengünstigsten Standorts bei gegebener Nachfrage. VI, 142 Seiten 1976
- Vol. 126: Energy, Regional Science and Public Policy Energy and Environment I. Proceedings 1975 Edited by M. Chatterji and P. Van Rompuy VIII, 316 pages 1976
- Vol 127: Environment, Regional Science and Interregional Modeling Energy and Environment II Proceedings 1975 Edited by M. Chatterji and P. Van Rompuy IX, 211 pages 1976.
- Vol. 128: Integer Programming and Related Areas. A Classified Bibliography. Edited by C. Kastning XII, 495 pages 1976
- Vol. 129. H.-J. Luthi, Komplementaritäts- und Fixpunktalgorithmen in der mathematischen Programmierung. Spieltheorie und Ökonomie. VII, 145 Seiten 1976
- Vol 130. Multiple Criteria Decision Making, Jouy-en-Josas, France. Proceedings 1975 Edited by H. Thirrez and S. Zionts. VI, 409 pages 1976.
- Vol 131: Mathematical Systems Theory Proceedings 1975 Edited by G. Marchesini and S. K. Mitter X, 408 pages 1976
- Vol 132 U. H. Funke, Mathematical Models in Marketing A Collection of Abstracts. XX, 514 pages 1976.
- Vol 133. Warsaw Fall Seminars in Mathematical Economics 1975 Edited by M. W. Łoś, J. Łoś, and A. Wieczorek. V. 159 pages 1976
- Vol 134 Computing Methods in Applied Sciences and Engineering Proceedings 1975. VIII, 390 pages 1976
- Vol 135 H. Haga, A Disequilibrium - Equilibrium Model with Money and Bonds. A Keynesian - Walrasian Synthesis VI, 119 pages 1976.
- Vol. 136: E. Kofler und G. Menges, Entscheidungen bei unvollständiger Information. XII, 357 Seiten 1976.
- Vol 137 R. Wets, Grundlagen Konvexer Optimierung VI, 146 Seiten 1976
- Vol. 138 K. Okuguchi, Expectations and Stability in Oligopoly Models VI, 103 pages 1976
- Vol 139. Production Theory and Its Applications Proceedings Edited by H. Albach and G. Bergendahl VIII, 193 pages 1977
- Vol 140 W. Eichhorn and J. Voeller, Theory of the Price Index. Fisher's Test Approach and Generalizations VII, 95 pages. 1976
- Vol 141: Mathematical Economics and Game Theory Essays in Honor of Oskar Morgenstern Edited by R. Henn and O. Moeschlin XIV, 703 pages 1977
- Vol 142 J. S. Lane, On Optimal Population Paths V, 123 pages 1977
- Vol. 143: B. Näsälund, An Analysis of Economic Size Distributions XV, 100 pages 1977
- Vol 144: Convex Analysis and Its Applications Proceedings 1976 Edited by A. Auslender VI, 219 pages 1977.
- Vol 145: J. Rosenmüller, Extreme Games and Their Solutions. IV, 126 pages. 1977.
- Vol 146. In Search of Economic Indicators Edited by W. H. Strigel XVI, 198 pages 1977
- Vol 147. Resource Allocation and Division of Space Proceedings. Edited by T. Fujii and R. Sato. VIII, 184 pages. 1977.
- Vol. 148: C. E. Mandl, Simulationstechnik und Simulationsmodelle in den Sozial- und Wirtschaftswissenschaften IX, 173 Seiten 1977.
- Vol. 149 Stationäre und schrumpfende Bevölkerungen: Demographisches Null- und Negativwachstum in Österreich. Herausgegeben von G. Feichtinger. VI, 262 Seiten 1977
- Vol 150. Bauer et al., Supercritical Wing Sections III. VI, 179 pages 1977.

Ökonometrie und Unternehmensforschung

Econometrics and Operations Research

- Vol. I Nichtlineare Programmierung. Von H. P. Kunzi und W. Krelle unter Mitwirkung von W. Oettli. Vergriffen
- Vol. II Lineare Programmierung und Erweiterungen. Von G. B. Dantzig. Ins Deutsche übertragen und bearbeitet von A. Jaeger. – Mit 103 Abbildungen. XVI, 712 Seiten. 1966. Geb.
- Vol. III Stochastic Processes. By M. Girault. – With 35 figures. XII, 126 pages. 1966. Cloth
- Vol. IV Methoden der Unternehmensforschung im Versicherungswesen. Von K. H. Wolff. – Mit 14 Diagrammen. VIII, 266 Seiten. 1966. Geb.
- Vol. V The Theory of Max-Min and its Application to Weapons Allocation Problems. By John M. Danskin. – With 6 figures. X, 126 pages. 1967. Cloth
- Vol. VI Entscheidungskriterien bei Risiko. Von H. Schneeweiss. – Mit 35 Abbildungen. XII, 215 Seiten. 1967. Geb.
- Vol. VII Boolean Methods in Operations Research and Related Areas. By P. L. Hammer (Ivănescu) and S. Rudeanu. With a preface by R. Bellman. – With 25 figures. XVI, 329 pages. 1968. Cloth
- Vol. VIII Strategy for R & D: Studies in the Microeconomics of Development. By Th. Marschak, Th. K. Glennan Jr., and R. Summers. – With 44 figures. XIV, 330 pages. 1967. Cloth
- Vol. IX Dynamic Programming of Economic Decisions. By M. J. Beckmann. – With 9 figures XII, 143 pages. 1968. Cloth
- Vol. X Input-Output-Analyse. Von J. Schumann. – Mit 12 Abbildungen. X, 311 Seiten. 1968. Geb.
- Vol. XI Produktionstheorie. Von W. Wittmann. – Mit 54 Abbildungen. VIII, 177 Seiten. 1968 Geb.
- Vol. XII Sensitivitätsanalysen und parametrische Programmierung. Von W. Dinkelbach. – Mit 20 Abbildungen. XI, 190 Seiten. 1969. Geb.
- Vol. XIII Graphentheoretische Methoden und ihre Anwendungen. Von W. Knodel. – Mit 24 Abbildungen. VIII, 111 Seiten. 1969. Geb.
- Vol. XIV Praktische Studien zur Unternehmensforschung. Von E. Nievergelt, O. Müller, F. E. Schlaepfer und W. H. Landis. – Mit 82 Abbildungen. XII, 240 Seiten. 1970. Geb
- Vol. XV Optimale Reihenfolgen. Von H. Müller-Merbach. – Mit 45 Abbildungen. IX, 225 Seiten. 1970. Geb.
- Vol. XVI Preispolitik der Mehrproduktenunternehmung in der statischen Theorie. Von R. Selten. – Mit 20 Abbildungen. VIII, 195 Seiten. 1970. Geb.
- Vol. XVII Information Theory for Systems Engineers. By L. P. Hyvarinen. – With 42 figures. VIII, 197 pages. 1970. Cloth
- Vol. XVIII Unternehmensforschung im Bergbau. Von F. L. Wilke. – Mit 29 Abbildungen. VIII, 150 Seiten 1972. Geb.
- Vol. XIX Anti-Aquilibrium. Von J. Kornai – Mit 31 Abbildungen. XVI, 382 Seiten. 1975. Geb.
- Vol. XX Mathematische Optimierung. Von E. Blum, W. Oettli. IX, 413 Seiten. (davon 75 Serten Bibliographie). 1975. Geb.
- Vol. XXI Stochastic Linear Programming. By P. Kall. VI, 95 pages. 1976. Cloth.

This series aims to report new developments in (mathematical) economics, econometrics, operations research, and mathematical systems, research and teaching – quickly, informally and at a high level. The type of material considered for publication includes:

1. Preliminary drafts of original papers and monographs
2. Lectures on a new field, or presenting a new angle on a classical field
3. Seminar work-outs
4. Reports of meetings, provided they are
 - a) of exceptional interest and
 - b) devoted to a single topic.

Texts which are out of print but still in demand may also be considered if they fall within these categories.

The timeliness of a manuscript is more important than its form, which may be unfinished or tentative. Thus, in some instances, proofs may be merely outlined and results presented which have been or will later be published elsewhere. If possible, a subject index should be included. Publication of Lecture Notes is intended as a service to the international scientific community, in that a commercial publisher, Springer-Verlag, can offer a wider distribution to documents which would otherwise have a restricted readership. Once published and copyrighted, they can be documented in the scientific literature.

Manuscripts

Manuscripts should comprise not less than 100 and preferably not more than 500 pages. They are reproduced by a photographic process and therefore must be typed with extreme care. Symbols not on the typewriter should be inserted by hand in indelible black ink. Corrections to the typescript should be made by pasting the amended text over the old one, or by obliterating errors with white correcting fluid. Authors receive 75 free copies and are free to use the material in other publications. The typescript is reduced slightly in size during reproduction; best results will not be obtained unless the text on any one page is kept within the overall limit of 18x26.5 cm (7x10½ inches). The publishers will be pleased to supply on request special stationery with the typing area outlined.

Manuscripts in English, German or French should be sent directly to Springer-Verlag New York or Springer-Verlag Heidelberg

Springer-Verlag, Heidelberger Platz 3, D-1000 Berlin 33
Springer-Verlag, Neuenheimer Landstraße 28–30, D-6900 Heidelberg 1
Springer-Verlag, 175 Fifth Avenue, New York, NY 10010/USA

ISBN 3-540-08533-5
ISBN 0-387-08533-5